User's Manual

Your Gateway to Machine Vision

Microsoft, Windows, Windows NT, Windows 2000, Windows XP, Visual Basic, Microsoft .NET, Visual C++, Visual C#, and ActiveX are either trademarks or registered trademarks of Microsoft Corporation.

All other nationally and internationally recognized trademarks and tradenames are hereby recognized.

Copyright © 2000-2008 by MVTec Software GmbH, München, Germany

Edition 1	November 2000	(ActivVisionTools 1.0)
Edition 2	Februar 2001	(ActivVisionTools 1.2)
Edition 3	April 2001	(ActivVisionTools 1.3)
Edition 4	September 2001	(ActivVisionTools 2.0)
Edition 5	November 2002	(ActivVisionTools 2.1)
Edition 6	January 2005	(ActivVisionTools 3.0)
Edition 7	February 2006	(ActivVisionTools 3.1)
Edition 8	May 2008	(ActivVisionTools 3.2)

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

More information about ActivVisionTools can be found at:

`http://www.activ-vision-tools.com`

How to Read This Manual

This manual explains how to use ActivView, the *basic tool* of all ActivVisionTools applications. It describes the functionality of ActivView with Visual Basic examples. Before reading this manual, we recommend to read the manual Getting Started with ActivVisionTools, which introduces the basic concepts of ActivVisionTools.

For each example in this manual, there is a corresponding Visual Basic project; these projects can be found in the subdirectory `examples\manuals\activview` of the ActivVisionTools base directory you selected during the installation (default: `C:\Program Files\MVTec\ActivVisionTools`). Of course, you can also create your own Visual Basic projects from scratch. Note that the screenshots in the manual may differ slightly from the corresponding Visual Basic projects. To follow the examples actively, first install and configure ActivVisionTools as described in the manual Getting Started with ActivVisionTools.

We recommend to **create a private copy of the example projects** because by experimenting with the projects, you also change their *state*, which is then automatically stored in the so-called description files (extension `.dsc`) by ActivVisionTools. Of course, you can restore the state of a project by retrieving the corresponding description file from the CD.

Contents

1 About ActivView **1**

1.1 Introducing ActivView 2

1.2 The Sub-Tools of ActivView 4

1.3 A Closer Look at Image Acquisition 8

2 Using ActivView **15**

2.1 Reading Images From Files 16

2.2 Grabbing Images 18

2.3 Inspecting and Modifying Properties of ActivVisionTools 20

2.4 Inspecting Images 22

2.5 Calibrating Image Coordinates 24

3 Combining ActivView with other ActivVisionTools **27**

3.1 Managing Regions of Interest 28

3.2 Displaying ROIs and Results 30

3.3 Inspecting the Status of an Application 32

3.4 Configuring the Two Execution Modes 34

3.5 Inspecting and Modifying Connections Between ActivVisionTools 36

4 Tips & Tricks **39**

4.1 Analyzing Image Quality 40

4.2 Troubleshooting 48

A Accessing Your Image Acquisition Device **51**

A.1 Supported Image Acquisition Devices 53

A.2 More Information About the Supported Image Acquisition Devices 53

A.3 Detection of Image Acquisition Devices 55

A.4 Modifying the List of Available Image Acquisition Devices 56

A.5 Setting Additional Parameters 57

A.6 How to Access an Unsupported Image Acquisition Device 64

Chapter 1

About ActivView

This chapter will introduce you to the features and the basic concepts of ActivView. It gives an overview about ActivView's *master tool* and its *support tools*, which are described in more detail in chapter 2 on page 15 and chapter 3 on page 27.

In section 1.3 on page 8 we take a closer look at the process of image acquisition, as this process is of fundamental importance to all machine vision applications.

1.1 Introducing ActivView **2**
1.2 The Sub-Tools of ActivView **4**
1.3 A Closer Look at Image Acquisition **8**
1.3.1 From the Camera into ActivView 8
1.3.2 About Images Themselves 10
1.3.3 The Question of Timing 12
1.3.4 Using Multiple Cameras 14

1.1 Introducing ActivView

ActivView is a *basic tool* of the ActivVisionTools family, i.e., this tool is part of every ActivVisionTools application. Furthermore, it is the only ActivVisionTool that can be used standalone. It provides the basic functionality necessary in each machine vision task: input of images (monochrome and color), management of regions of interest, and display of images and results. Furthermore, ActivView includes components for calibrating the size of a pixel.

Besides machine vision functionality, ActivView provides components for switching between the two execution modes, the *configuration mode* and the *application mode*, for customizing the appearance of an ActivVisionTools application in the two modes, and for displaying the current status of an application. Furthermore, it allows to inspect and modify the connections between ActivVisionTools and their properties interactively.

Image Input

With ActivView, you can input images both from image acquisition devices and from files. Up to now, interfaces to about 50 image acquisition devices are integrated in ActivView. With the help of these interfaces, you can grab an image within a few mouse clicks. If supported by the image acquisition device itself, you can easily couple the grabbing of an image to an external trigger signal.

Images can also be read from files, in various formats (see section 2.2 on page 18 for a complete list). Furthermore, you can create and read sequences of images.

ActivView will automatically send acquired images to all connected ActivVisionTools, you do not need to write a single line of code.

Regions of Interest (ROIs)

In order to speed up image processing, it is useful to restrict it to so-called *regions of interest* (ROIs) in the image. With ActivView, you can specify such ROIs for each subsequent *vision tool* (and *primary vision tool*) interactively: You select the tool, choose a shape (ellipse, rectangle, etc.), and place the ROI with the mouse on the image. ActivView will then automatically send only the corresponding part of the acquired images to the selected ActivVisionTool.

It is possible to assign different ROIs to different ActivVisionTools; furthermore, you can create multiple ROIs for one tool.

Display of Images, ROIs, and Results

ActivView provides a central *display* for the visualization of images, ROIs, and results. All connected ActivVisionTools automatically send their results back to ActivView to be overlayed

on the current image, without any action required on your side. With the help of `AVTView-DisplayModes`, one of the *support tools* of ActivView, you can adapt the visualization to your needs (colors, drawing mode, line width, etc. – see section 3.2 on page 30).

For a closer inspection of images, ROIs, and results, ActivView provides a zooming mechanism. Moreover, the three *inspection tools* ActivZoom, ActivHistogram, and ActivLineProfile are provided together with ActivView.

Calibration of the Size of a Pixel

ActivView provides an easy to use, interactive mechanism to calibrate the size of a pixel in a plane perpendicular to the camera. If this mechanism is used, all subsequent measurements are converted automatically from pixels to other units.

Note that this calibration is not very accurate; for high-accuracy 3D measurements, please use ActivGeoCalib.

Switching Between Execution Modes

In an ActivVisionTools application, there are two execution modes: the *configuration mode* and the *application mode*. This allows you to create two versions of your machine vision application: one for its initial setup and configuration, and one for the final, running system. ActivView provides a *support tool* for customizing the appearance of an application in the two modes, and for switching between them. Furthermore, another support tool provides a single button to start and stop an application.

Inspecting the Status of an Application

The *support tool* `AVTViewStatus` displays the current status of your ActivVisionTools application, together with error messages. Furthermore, this tool shows the time needed for the last processing cycle, or alternatively the current "speed" of the application, i.e., the number of processing cycles per second.

Inspecting and Modifying Connections Between ActivVisionTools

ActivVisionTools automatically creates the connections between the individual tools. With the *support tool* `AVTViewConnections`, you can inspect and modify the connections.

Inspecting and Modifying Properties of ActivVisionTools

With the *support tool* `AVTViewProperties`, you can inspect and modify the properties of the ActivVisionTools in your application at run time.

1.2 The Sub-Tools of ActivView

Besides its *master tool*, ActivView provides eleven *support tools* (see figure 1.1 on page 5 and figure 1.2).

`AVTView` is the *master tool* of ActivView. It provides the central display for images, ROIs, and results. You can zoom into the display and move the part of the image that is displayed. How to use `AVTView` is described in more detail in section 2.4 on page 22.

`AVTViewFG` is a *support tool* of ActivView. With it, you can acquire images from image acquisition devices or from files and also save them to disk. How to use `AVTViewFG` is described in section 2.2 on page 18.

`AVTViewFGParameter` is a *support tool* of ActivView (not shown in figure 1.1). With it, you can set special parameters of your image acquisition device interactively. How to use `AVTViewFGParameter` is described in section A.5 on page 57.

`AVTViewStatus` is a *support tool* of ActivView. It displays the current status of an ActivVisionTools application. How to use and configure `AVTViewStatus` is described in section 3.3 on page 32.

`AVTViewROI` is a *support tool* of ActivView. It enables you to assign regions of interest to other ActivVisionTools. How to use `AVTViewROI` is described in section 3.1 on page 28.

`AVTViewDisplayModes` is a *support tool* of ActivView. It allows to change the way results and ROIs are displayed. How to use `AVTViewDisplayModes` is described in section 3.2 on page 30.

`AVTViewCalibration` is a *support tool* of ActivView. By drawing a line-shaped ROI in the image and specifying its length in a certain unit, you can convert pixel coordinates into this unit. How to use `AVTViewCalibration` is described in section 2.5 on page 24.

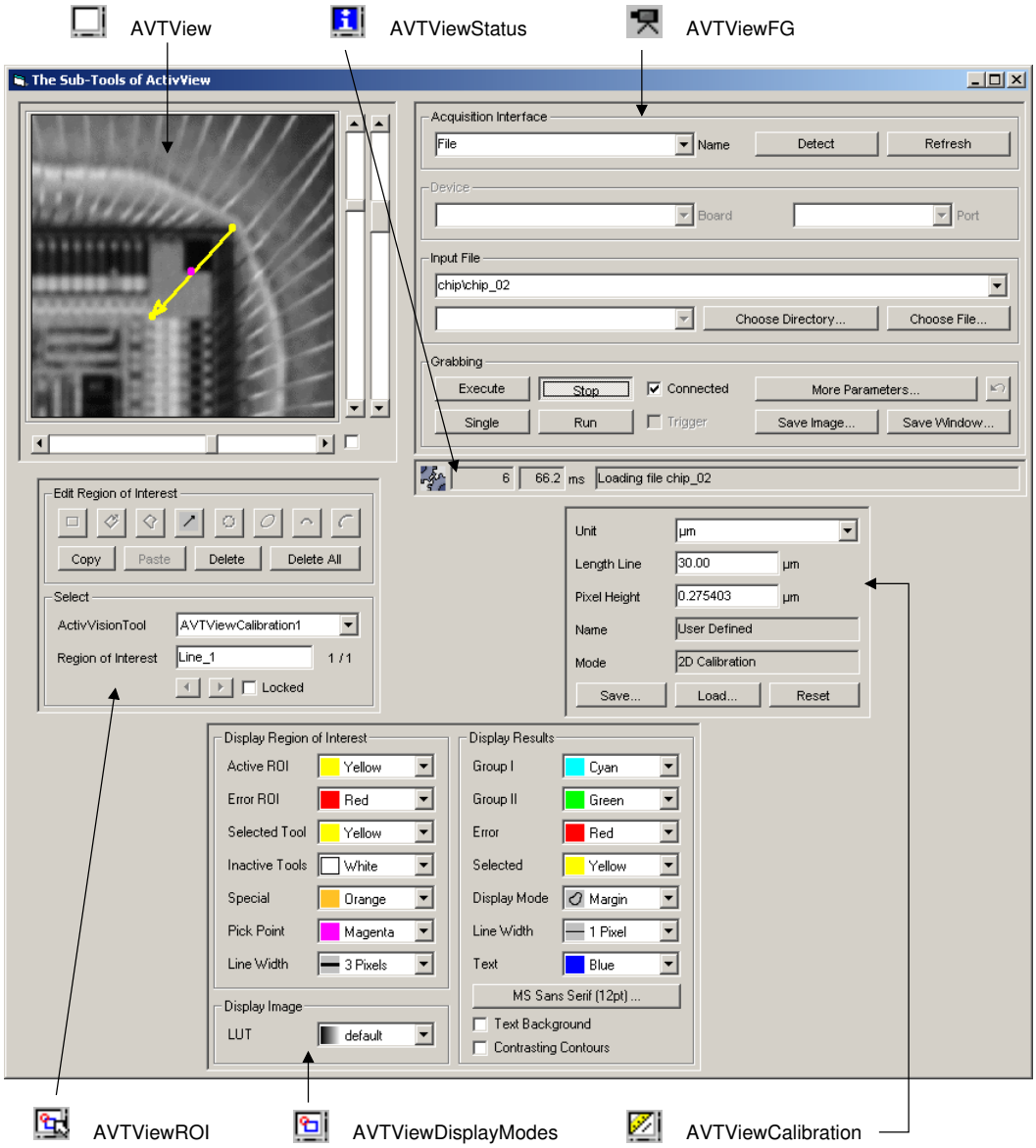


Figure 1.1: The sub-tools of ActivView (part 1).

`AVTViewExecute` is a *support tool* of ActivView. It allows to switch between *configuration mode* and *application mode*. How to use `AVTViewExecute` is described in section 3.4 on page 34.

`AVTViewExecuteSimple` is a *support tool* of ActivView. It provides a button to start and stop your application. How to use `AVTViewExecuteSimple` is described in section 3.4 on page 34.

`AVTViewConfigExec` is a *support tool* of ActivView. It allows to configure what is displayed in the different execution modes. How to use `AVTViewConfigExec` is described in section 3.4 on page 34.

`AVTViewConnections` is a *support tool* of ActivView. With it, you can inspect and modify the how tools are connected to each other. How to use `AVTViewConnections` is described in section 3.5 on page 36.

`AVTViewProperties` is a *support tool* of ActivView. It allows to inspect and modify the properties of the ActivVisionTools in an application. How to use `AVTViewProperties` is described in section 2.3 on page 20.

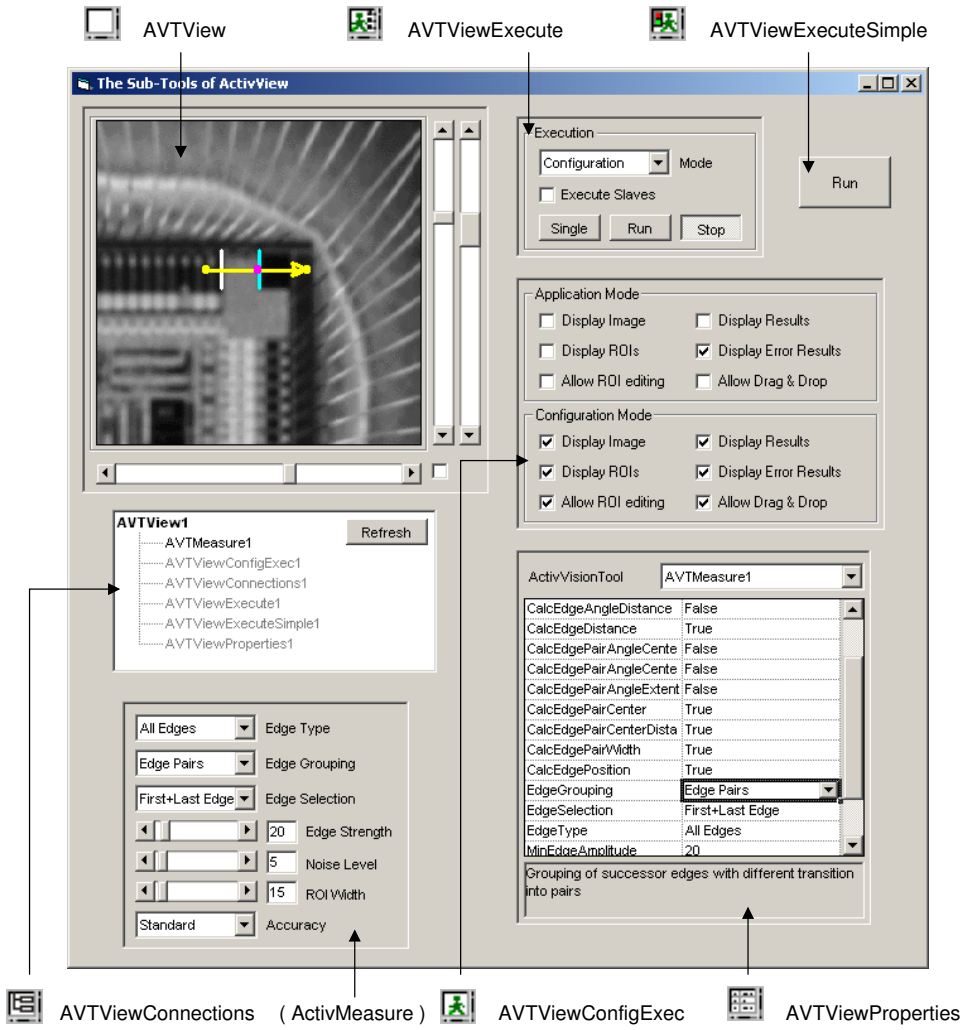


Figure 1.2: The sub-tools of ActivView (part 2).

1.3 A Closer Look at Image Acquisition

The process of image acquisition is the fundament of every machine vision application. To continue this analogy, one would like to set up this fundament only once and for well, and then be able to concentrate on building the application itself. Following the ActivVisionTools paradigm, ActivView actively supports you in setting up image acquisition, e.g., by interfacing automatically to the software provided by manufacturers of image acquisition devices and by limiting the remaining parameterization to the absolute necessary.

However, ActivView cannot determine all parameters automatically, especially if you are using a more complex image acquisition device with more capabilities. To facilitate the remaining parameterization effort, the following sections give a quick overview on how images get from the camera into ActivView, describe the different types of images, and take a look at the question when exactly an image is acquired.

1.3.1 From the Camera into ActivView

For the acquisition of images, multiple components have to interact and cooperate (see figure 1.3): A camera, in many cases a *frame grabber*, and the host computer, or more exactly software running on it.

① The Camera

Besides offering a different performance or functionality, cameras may differ significantly regarding their output. In the case of an analog output, a small number of standards predominates (e.g. NTSC or PAL). Typical digital output standards are IEEE 1394 (FireWire) and CameraLink, with Gigabit Ethernet (GigE) in the coming.

② The Frame Grabber

The main task of a frame grabber is to get the acquired images into the computer in a form which can be “understood” by the software. Besides this, a frame grabber can also be used to control the camera, with possible activities ranging from starting the next grab to more complex ones like reconfiguring the camera to grab only a certain part of the image. Most frame grabbers allow to connect them to an external trigger signal which then triggers the next grab (see section 1.3.3 on page 12).

With some cameras, you don’t need a dedicated frame grabber board. Instead, its task is performed by a standard adapter in the PC, e.g., a USB, FireWire, or Ethernet adapter. Please note that it is not sufficient that a camera has a USB, IEEE 1394, or Ethernet output in order to get its images into the PC; you also need the corresponding driver software (see below)!

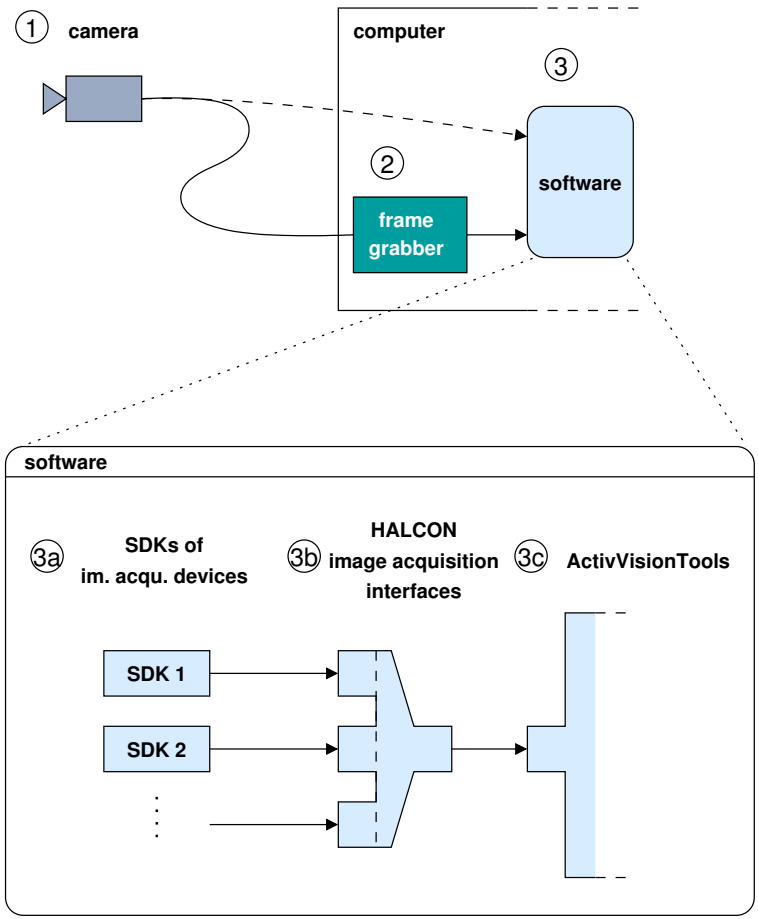


Figure 1.3: The components of the image acquisition process.

③ The Software

On the software side, again multiple components cooperate to get images inside ActivVisionTools. First of all, most frame grabbers come with their own software, the so-called SDK (③a, SDK = software development kit) consisting of the so-called *device driver* and one or more libraries (DLLs). The SDK performs the low-level communication with its frame grabber, e.g., manages its initialization, sends commands and receives and stores the images in main memory, and encapsulates it behind an easier-to-use interface. Note that also image acquisition devices that can be connected directly to a computer bring their own SDK.

However, each SDK presents a different interface. To facilitate the use of different image acquisition devices in a machine vision application, HALCON, the basis of the ActivVisionTools, therefore provides the so-called *image acquisition interfaces* (3b). These software components "translate" the commands and methods used by the corresponding SDK into a common, generic interface, which is then accessed by the ActivVisionTools (3c), or more exactly by `AVTViewFG`.

1.3.2 About Images Themselves

Figure 1.4 visualizes some technical terms concerning images and their representation.

① Color Images and the Concept of Channels

While a so-called gray value image, also called black-and-white image, contains only information about the brightness (also called intensity) of a certain object, color images also code its color. There exist many different ways to represent color. In the so-called *RGB color space* colors are represented by the sum of the three principal colors red, green, and blue.

Therefore, one can think of an RGB image as being composed by three sub-images called *channels*: The red channel contains all the red parts of an image (in the example in figure 1.4 the word 'RED'), the green channel all the green parts (the word 'GREEN'), and the blue channel all the blue parts (the word 'BLUE'). All other colors can be created by "mixing" the three principal colors. For example, the color cyan is composed by adding green and blue, in the example the word 'CYAN' therefore appears both in the green and in the blue channel.

! Please note that in the current version of ActivVisionTools color images are displayed including color, but **image proccessing is performed only on the red channel**. Via the programming interface, however, you can let the tools process on of the other channels instead; see the Advanced User's Guide for ActivVisionTools, section 3.4.2.1 on page 76 or section 3.6.1.3 on page 96, for examples.

② – ③ Frames vs. Fields, ④ – ⑤ Interlacing vs. Progressive Scan (relevant only for analog cameras)

A *frame* corresponds to one image out of a sequence of images. In analog standards PAL and NTSC, a frame is divided into two *fields*: The *even field* contains all lines with even numbers (0, 2, ...), the *odd field* those with odd numbers (1, 3, ...). Instead of the frames, only fields are grabbed and transmitted alternatingly, and then merged (*interlaced*) to form a full frame again.

If a fast moving object is imaged, interlacing leads to artefacts, which degrade image quality significantly as shown in figure 1.4. Therefore, the so-called *progressive scan* mode is more appropriate: Here, frames are grabbed and transmitted without dividing them into fields. Note that if a camera offers both interlacing and progressive scan, choosing progressive scan does not mean that you get more frames per second, only that a full frame is grabbed at a time.

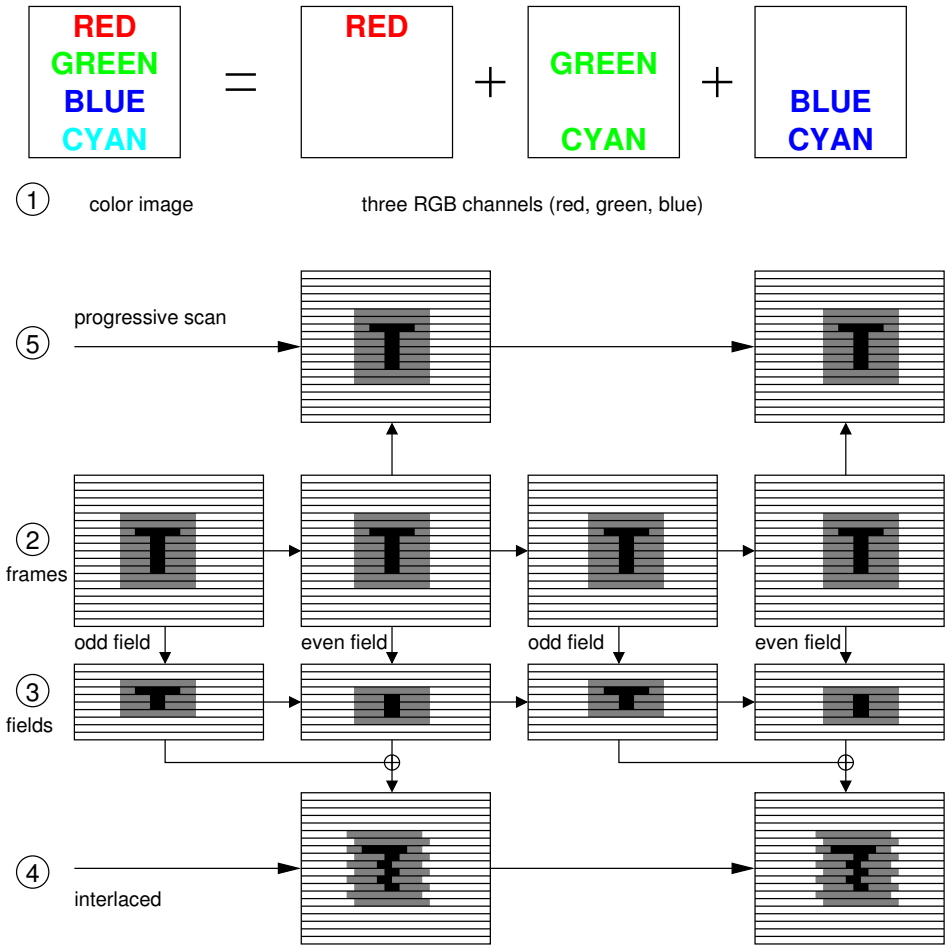


Figure 1.4: Fields, frames, and channels.

Image Resolution

Image resolution concerns two aspects: In measuring applications for example, the effective size of a pixel (i.e., the size of what is projected onto a pixel) is important as it influences the accuracy of measuring. This *spatial image resolution* has to be kept in mind when setting up the vision system for a machine vision application.

A second aspect concerns the *resolution of information*, in other words, how many bits are used for each pixel, e.g., 8 bit for gray value images or $3 \times 8 = 24$ bit for RGB images. Nowadays

many cameras offer other resolutions, e.g., 10 or 12 bit gray value images. The higher the resolution, the better objects can be differentiated by their appearance in the image. This is important for example in blob analysis applications, as objects (blobs) can be extracted only if they differ significantly from the background. Note that the pixel value resolution may be smaller than the number of bits actually transmitted (and stored) per pixel, as the latter usually is a multiple of 8.

! Please note that in the current version of ActivVisionTools **many tools can only process 8 bit images**, e.g., ActivBarcode or ActivAlignment; ActivView, however, can display images regardless of their resolution.

1.3.3 The Question of Timing

As multiple components cooperate to grab images the question of timing arises. Here, methods range from purely sequential to externally triggered grabbing. Figure 1.5 visualizes some of them.

① Sequential Grabbing

The naïve approach is to let components work in sequence: A cycle starts with the software requesting the next image. Typically, the image acquisition device must wait for the beginning of the next frame before starting to grab it. Only after the image is fully grabbed, the software can start with the actual processing. As can be seen in figure 1.5, even if the processing takes less than the grabbing not all images can be processed.

② Parallel Grabbing

To optimize the performance, ActivVisionTools therefore internally parallelizes the grabbing and the actual processing. The main idea is that while an image is processed, the image acquisition device already grabs the next image. In the example shown in figure 1.5 every frame can be processed this way.

③ Externally Triggered Grabbing

In many machine vision applications image grabbing must be synchronized with a physical process, e.g., by using photoelectric barriers that report when a part has arrived for inspection. Most image acquisition devices allow to connect them to such an *external trigger* signal. From the point of view of an ActivVisionTools application, this mode works like parallel grabbing, with the only difference that now the image acquisition device must wait for the trigger before it grabs the next image.

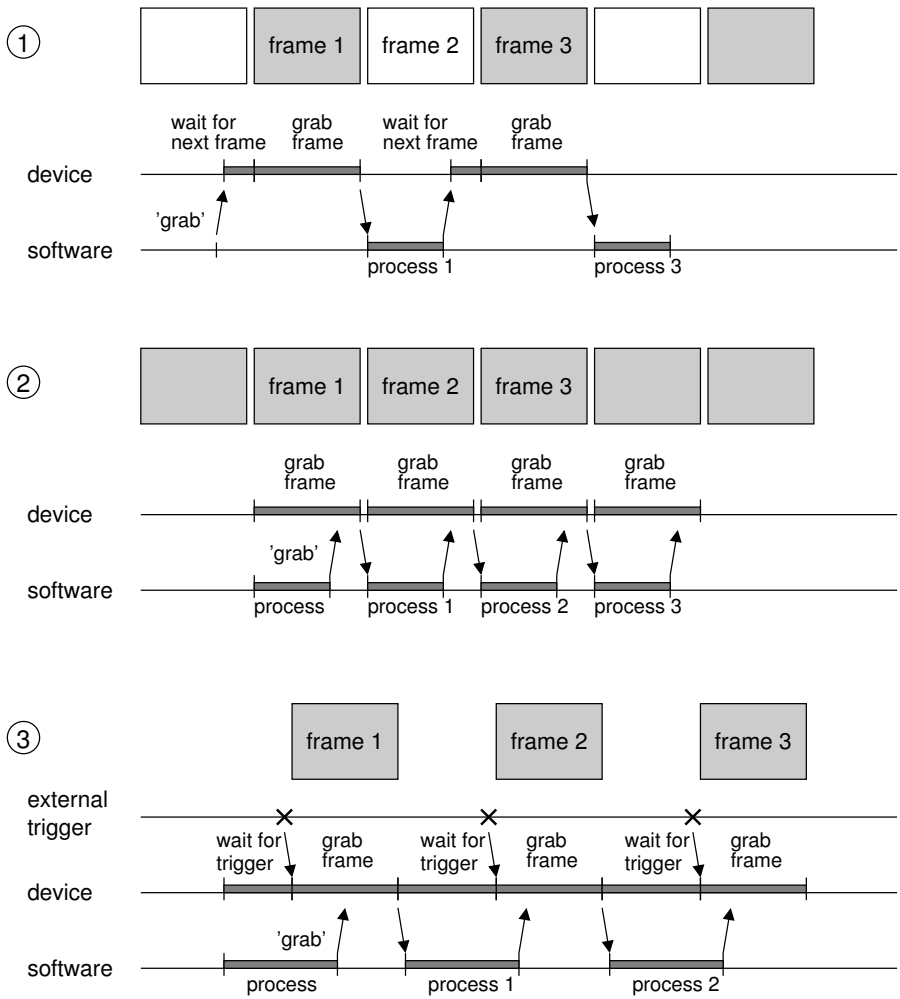


Figure 1.5: ① sequential grabbing mode; ② parallel grabbing (ActivVisionTools default mode); ③ externally triggered parallel grabbing.

Real-Time Machine Vision

As a technical term, the attribute *real-time* means that a process guarantees that it meets given deadlines. In a machine vision application, real-time behavior may be required both for the process of image grabbing (e.g., an image must be grabbed while the part to be inspected is completely visible) and for the actual image processing in order to be ready when the next part,

i.e., the next image, arrives.

Please keep in mind that **Windows is no real-time operating systems**. This means that the operating system itself does not guarantee that your process will get the necessary processing time before its deadline expires. In other words, all efforts for real-time behavior are already thwarted at the basis.

From the point of view of a machine vision application running under Windows, the most you can do is assure that real-time behavior is not already prevented by the application itself. ActivVisionTools supports this by parallelizing image grabbing and processing and by allowing to switch off various display and output activities in the *application mode*.

If your application has hard real-time requirements concerning image grabbing, there is one thing to note on the use of an external trigger: In order to really synchronize the grabbing process with an external event, the camera must be able to start grabbing a new image immediately as depicted in diagram (3) of figure 1.5 on page 13 . This feature is typically called *asynchronously resettable*. If a camera does not have this feature, the system must wait for the camera to finish its current frame before starting the desired grab. This means that there will be a delay between the external trigger and the start of the grabbing which may lie between zero and the time needed for one frame.

1.3.4 Using Multiple Cameras

You can acquire images from multiple cameras by using multiple instances of `AVTView`. For detailed information please refer to the Advanced User's Guide for ActivVisionTools, section 3.6 on page 88.

Chapter 2

Using ActivView

This chapter will explain those parts of ActivView that can be used stand-alone, i.e., without other ActivVisionTools: how to acquire and inspect images and how to calibrate image coordinates. Furthermore, it describes how to modify properties of ActivVisionTools.

2.1 Reading Images From Files 16
2.2 Grabbing Images 18
2.3 Inspecting and Modifying Properties of ActivVisionTools 20
2.4 Inspecting Images 22
2.5 Calibrating Image Coordinates 24

2.1 Reading Images From Files Using

Using AVTViewFG, you can input images that have been stored as files. Currently, the following formats are supported: BMP, PNG, GIF, TIFF, JPEG, PNM, PCX, XWD, AVI, and binary.

Furthermore, you can read sequences of images by using a special file format (extension .seq). Instead of using sequence files, you can also load all images in a directory consecutively. The images are loaded sorted by name. Subdirectories are not scanned.

Visual Basic Example

Preparation for the following example:

- ☐ Open the project imagefiles\view_imagefiles.vbp. Alternatively, create a new project and place AVTView, AVTViewFG, and AVTViewStatus on the form by double-clicking the icons , , and , respectively, with the left mouse button.
- ☐ Execute the application (Run ▷ Start or via the corresponding button).

The following steps are visualized in figure 2.1.

① Switch the image input mode to 'File' in the combo box Name. AVI files can also be loaded in the mode 'DirectFile'.

② In the frame Input File, you can now select among a list of image files provided with the ActivVisionTools CD from the combo box, or open a file selector by clicking Choose File or Choose Directory. The image is loaded and displayed when you press the Enter key.

③ Alternatively, you can click Single to load (or reload) an image. If you have selected an image directory or a sequence, e.g. pendulum\pendulum.seq, at each click the next image will be loaded.

④ Click Execute to re-execute all tools on the current image. This is just like clicking Single without skipping to the next image. This button is useful if you want to evaluate the effect of changed parameters on the current image.

⑤ If you click Run, images are loaded in a loop. If you have selected a single file, it will be reloaded continuously (this does not make much sense). You stop this mode by clicking Stop.

⑥ AVTViewStatus displays error messages, e.g., if you try to load a not existing image file. If you grab from an image sequence, it displays the current file name.

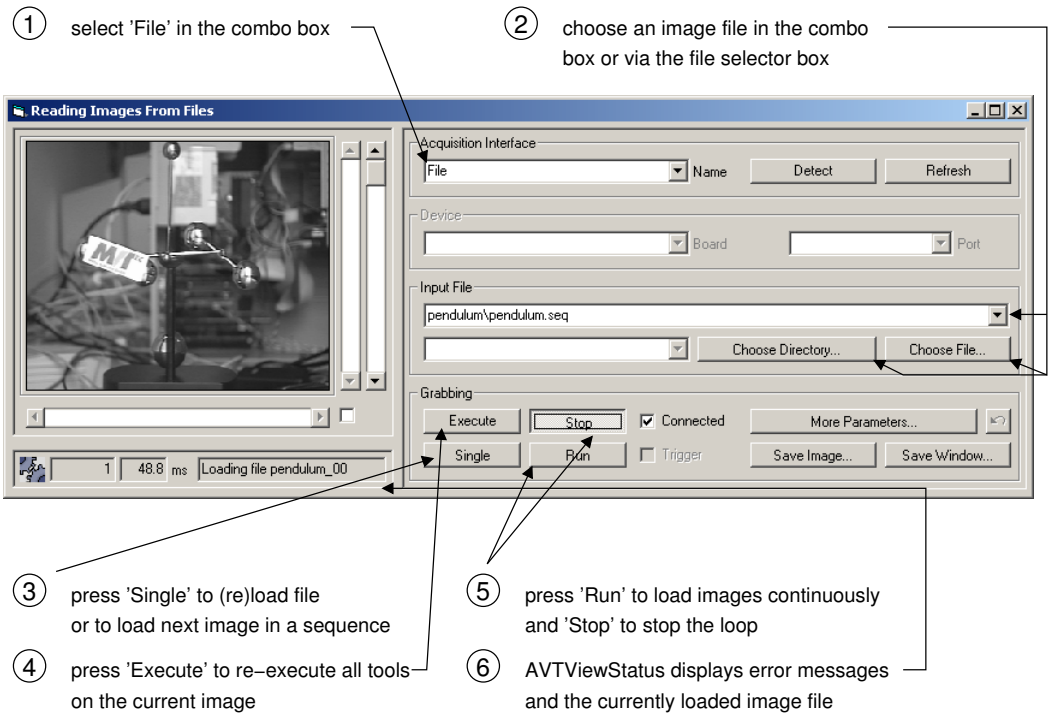


Figure 2.1: Loading images from files.

We recommend to **adapt the size of the display to the size of the image** as described in section 2.3 on page 20.

2.2 Grabbing Images Using

Besides image input from files, `AVTViewFG` provides interfaces to more than 50 image acquisition devices (see section A.1 on page 53 for a complete list). Before accessing an image acquisition device from within ActivVisionTools, we recommend to check that it's working correctly. Suitable test programs should be provided by the manufacturer of the device.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project.
- ☐ Otherwise, open the project `framegrabber\view_framegrabber.vbp` or create a new project and place `AVTView`, `AVTViewFG`, and `AVTViewStatus` on the form by double-clicking the icons , , and , respectively, with the left mouse button.

  Execute the application (`Run` ▷ `Start` or via the corresponding button).

The following steps are visualized in figure 2.2.

1. Click `Detect` to update the list of available image acquisition devices. Select your image acquisition device in the combo box `Name`. If your device does not appear in the list, please refer to section 4.2 on page 48.
2. Depending on the used image acquisition device, you might need to specify additional parameters describing the device, e.g., the type of your frame grabber board or camera (combo box `Board`) or the number of the input (port) at which the camera is connected to the frame grabber or computer (combo box `Port`). Click `Refresh` to update the supported parameter values for the connected interface.
3. Further parameters describe the connected camera and the image format. In the combo box `CamType`, you can select the type of the analog video signal (e.g., `'ntsc'`) or the digital image format. For some image acquisition devices, it may be necessary to specify a camera configuration file. Please refer to section A.2 on page 53 for more information.

   In the combo box `Color`, specify the type of the image together with its pixel value resolution (see section 1.3.2 on page 10), e.g., `'Gray 8'` or `'Color 24'`.

   When using analog cameras, the combo boxes `Size` and `Field` let you you select the image size and the grabbing modesection 1.3.2 on page 10). Note that the option `'Half Height'` means that single fields are grabbed; which field is grabbed can be specified in `Field` (`'First'`: odd fields, `'Second'`: even fields, `'Next'`: both fields).
4. Enable or disable an external trigger with the check box ☑ `Trigger`.
5. If you click , all parameters are reset to their default values.

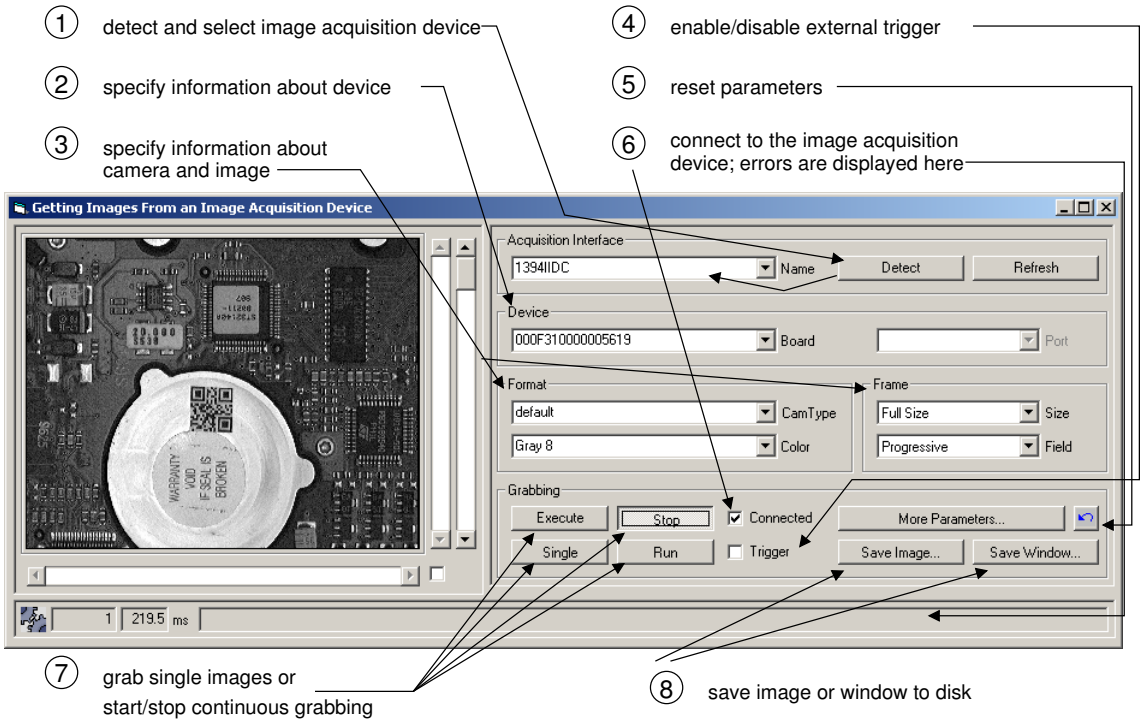


Figure 2.2: Grabbing images from a camera.

(6) Connect to your image acquisition device by checking ☑ `Connected`. Depending on the device, connecting may take some time. In case of problems, `AVTViewStatus` displays an error message (for troubleshooting see section 4.2 on page 48).

After connecting, you can set additional parameters by clicking `More Parameters` (see section A.5 on page 57).

(7) Grab single images by clicking `Single`. Re-execute the tools on the current image by clicking `Execute`. Start and stop continuous grabbing by clicking `Run` and `Stop`.

(8) To save a grabbed image to disk, click `Save Image`; in the appearing file selector box, specify a file name. To save only the displayed part of the image (e.g., when zoomed in), click `Save Window`.

Please note that in order to speed up the image display, you should **adapt the size of the display to the size of the image** as described in the next section. !

2.3 Inspecting and Modifying Properties of ActivVisionTools Via

Information about the ActivVisionTools is stored in their so-called *properties*. Properties can be divided in two classes: standard properties of ActiveX controls, e.g., `Width` or `Visible`, and properties that are special to ActivVisionTools, e.g., the parameter values you select in the ActivVisionTools controls and dialogs.

A second classification concerns the time when you can modify a property: Typically, (standard) layout properties like `Width` can only be modified at design time, either by interacting with the controls on the form or in the *Property Window*. In contrast, most ActivVisionTools properties can only be modified at run time, by selecting parameter values in the controls or via ActivView's support tool `AVTViewProperties`.

We illustrate the process of modifying properties in `AVTViewProperties` and the *Property Window* with the task of adapting the display to the image size.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on one of the previous examples, you may continue using this project. Otherwise, create a project containing `AVTView`.

The following steps are visualized in figure 2.3.

1. At run time, open `AVTViewProperties` by clicking on `AVTView` with the right mouse button and selecting `Properties` in the appearing context menu.
2. Select the ActivVisionTool whose properties you want to inspect in the combo box `ActivVisionTool`. Below, you can then inspect and modify the properties. At the bottom of the dialog a short help text describes the currently selected property.

   Please note that `AVTViewProperties` displays the actual name of the properties and not the (possibly translated) text that appears beside the corresponding GUI element!
3. To adapt the size of the display to the size of the image, first you must query the size of the image. For this, select `AVTView` in the combo box and inspect the value of the properties `LastImageWidth` and `LastImageHeight`.
4. Then stop the application. At design time, select `AVTView` by clicking on it. Its design-time properties are displayed in the *Property Window*.
5. The size of the display is controlled by the properties `HeightImageWindow` and `WidthImageWindow`. To speed up the display, we recommend to set these properties

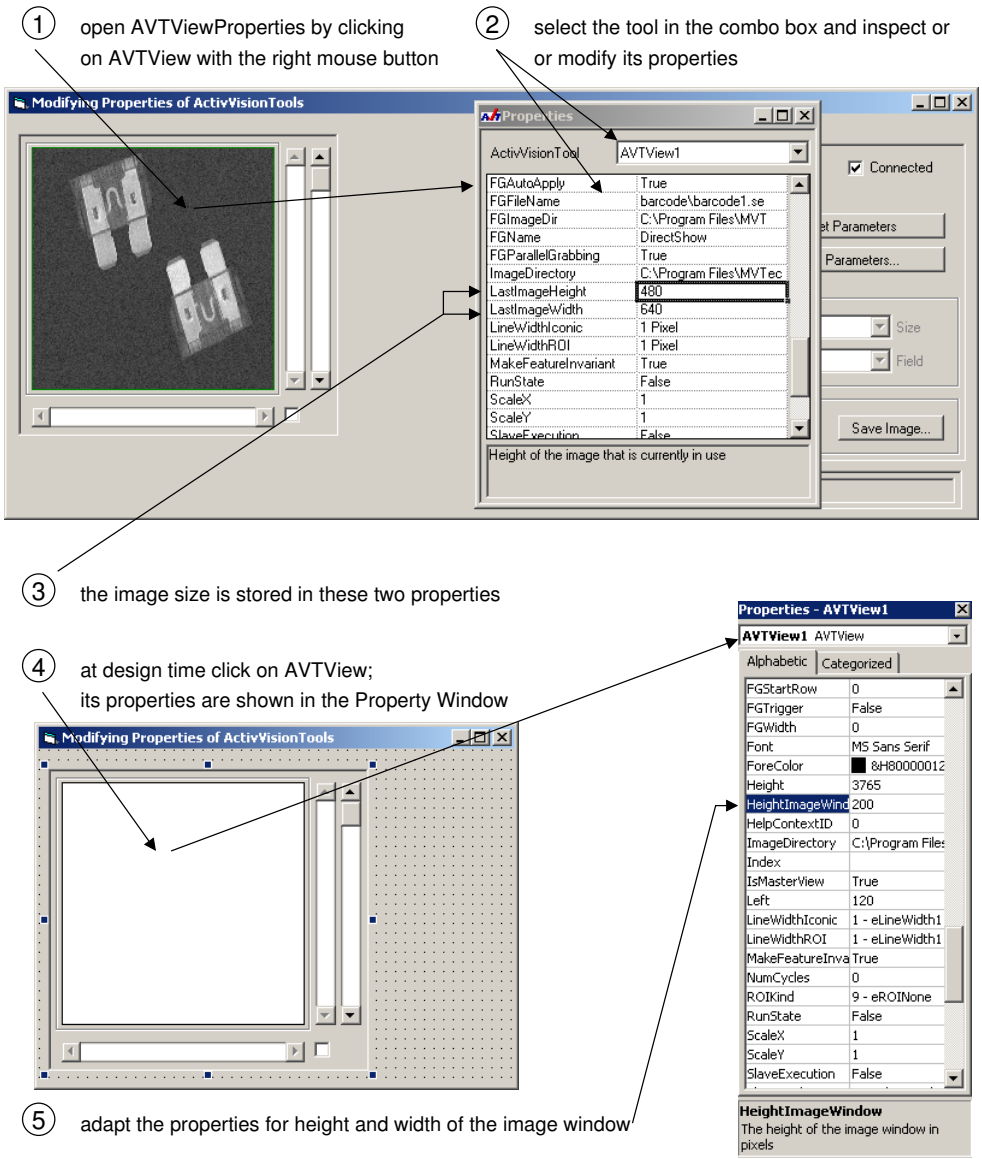


Figure 2.3: Modifying properties of ActivView to adapt the display to the image size.

to the full, half, or quarter size of the image.

2.4 Inspecting Images Using

`AVTView` allows to inspect images more closely by providing a zoom mechanism. The *inspection tools* ActivHistogram and ActivLineProfile allow a more detailed inspection; please refer to section 4.1.1 on page 40 for more information.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. Otherwise, open the project `display\view_display.vbp`.
- ☐ Execute the application (`Run` ▷ `Start` or via the corresponding button).

The following steps are visualized in figure 2.4.

① To zoom into the image, drag the outer scrollbar on the right side of the display. Note that the zooming factor does not increase continuously, but following a power series (2, 4, 8, 16, . . .) to speed up zooming.

② After zooming you can change the part of the image that is displayed by using the two scrollbars below and at the right side of the display.

③ If you check the box in the lower right corner, you can “move” in the image by moving the mouse inside the image while keeping the left mouse button pressed.

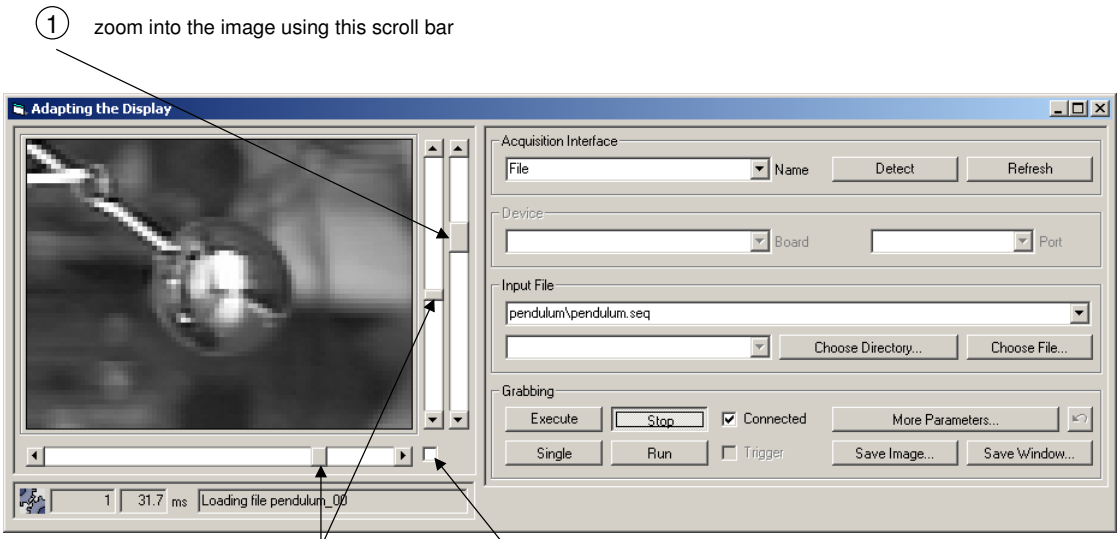


Figure 2.4: Zooming into the image.

2.5 Calibrating Image Coordinates Using

Using `AVTViewCalibration`, you can determine the size of a pixel (i.e., its height as square pixels are assumed) by drawing a line in the image and specifying its length.This information is passed on by ActivView to subsequent tools which use it to convert measurements from pixels to another unit. Note that this conversion only works if the measured objects lie in the same plane, i.e., at the same distance from the camera, as the line whose length was specified. For high-accuracy 3D calibration ActivGeoCalib should be used.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. Switch into design mode (`Run` ▷ `Stop` or via the corresponding button). Select `AVTViewFG` by clicking into it and delete it by pressing `Delete`. Add `AVTViewROI` and `AVTViewCalibration` to the form by double-clicking the icons and with the left mouse button.

  Otherwise, open the project `calibration\view_calibration.vbp`.

- ☐ Execute the application (`Run` ▷ `Start` or via the corresponding button). Here, you can open `AVTViewFG` by clicking on `AVTView` with the right mouse button and selecting `Image Acquisition`. Load the image sequence `pendulum\pendulum.seq`.

The following steps are visualized in figure 2.5.

1. Make sure that the check box at the lower right corner of the display is not checked. Otherwise, all mouse clicks in the image will move the displayed part of the image!
2. In the combo box `ActivVisionTool` of `AVTViewROI`, select `AVTViewCalibration1`.
3. Select the shape of the ROI by clicking the button .
4. To draw the line in the image, click into it with the left mouse button, keep the button pressed, and move the mouse. A line appears, with its endpoint marked by an arrow.
5. Edit it by dragging the pick points. To position the line accurately, zoom the image as described in the previous section.
6. In `AVTViewCalibration`, select a unit in the combo box `Unit`. This unit is automatically adopted both for the length of the line and for the pixel height.
7. Then, specify the length of the line you have drawn; in the example, the line drawn across the small card has a length of `3.7cm`. Press `Enter` to compute and display the height of a pixel.

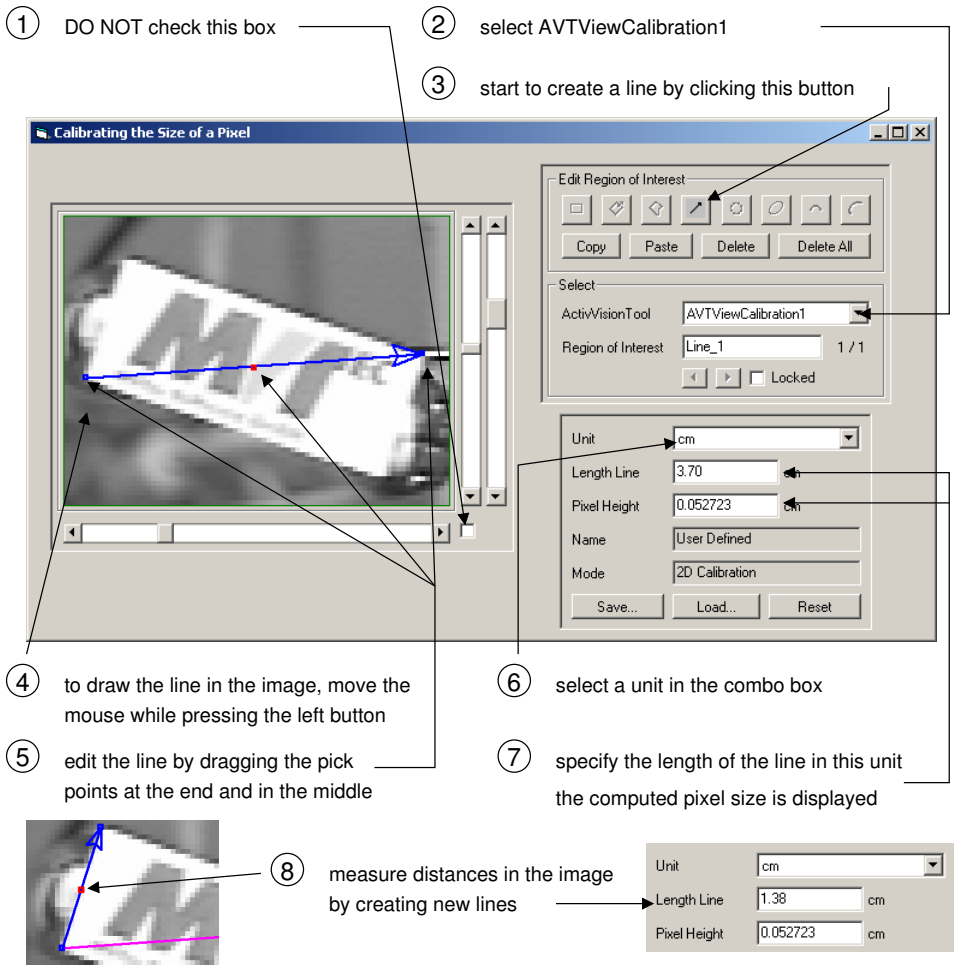


Figure 2.5: Calibrating the size of a pixel.

⑧ Now, you can measure distances in the image by creating new lines. Note that the calibration is only valid for the logo!

Note that if you know the pixel size already, you can specify it directly in the text box `Pixel Height` in `AVTViewCalibration`.

Chapter 3

Combining ActivView with other ActivVisionTools

This chapter explains the parts of ActivView that are used in conjunction with other tools: how to create and edit ROIs and how to change the display of ROIs and results. For demonstration, ActivBarcode is used as an example *vision tool*; please consult the User's Manual for ActivBarcode for detailed information about it. Note that if you do not have a license for this tool, an error message appears when you add it to the form, and the control remains disabled.

The final sections focus on ActivView's role as the central instance of an ActivVisionTools application. They describe the display of status information, the customization of the behavior in the two execution modes, and how to inspect and modify the connections between individual ActivVisionTools.

3.1 Managing Regions of Interest 28
3.2 Displaying ROIs and Results 30
3.3 Inspecting the Status of an Application 32
3.4 Configuring the Two Execution Modes 34
3.5 Inspecting and Modifying Connections Between ActivVisionTools 36

3.1 Managing Regions of Interest Using

Using `AVTViewROI`, you can specify one or more regions of interest (ROIs) per ActivVisionTool. Image processing will then be restricted to the corresponding parts of the image(s).

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example in chapter 2 on page 15, you may continue using this project. At design time, select `AVTViewCalibration` by clicking into it with the left mouse button and delete it by pressing `Delete`. Instead, add `AVTBarcode1D` to the form by double-clicking the icon .

  Otherwise, open the project `rois\view_rois.vbp`.

- ☐ Execute the application. Open `AVTViewFG` by clicking on `AVTView` with the right mouse button and selecting `Image Acquisition`. Load `barcode\ean13_08`.

The following steps are visualized in figure 3.1.

① Make sure that the check box at the lower right corner of the display is not checked. Otherwise, all subsequent mouse actions in the image will result in moving the displayed part of the image.

② In the upper combo box of `AVTViewROI`, all connected ActivVisionTools are displayed, in this case only `AVTBarcode1D1`, which is therefore automatically selected as the recipient of ROIs.

③ Not all ROI shapes are allowed for ActivBarcode. Select for example an arbitrarily-oriented rectangle by clicking .

④ To draw the rectangle in the image, click into it with the left mouse button, keep the button pressed, and move the mouse. A narrow rectangle appears, its orientation following the mouse. The creation of a polygonal ROI ( ) is more complex: By the first mouse movement, the first side of the polygon is created. You can add a new corner point by clicking on the line with the left mouse button; when you drag it by keeping the mouse button pressed, new polygon sides are created, where you can again add new corner points. To delete a corner point, drag it onto a neighboring corner point.

⑤ After releasing the mouse button, you can move the ROI by dragging its pick point in the middle and edit its shape by dragging the outer pick points. Note that for a polygonal ROI the “middle point” appears at the center of gravity of the ROI and therefore changes whenever the polygon is modified; besides, it may even lie outside the ROI for concave polygons.

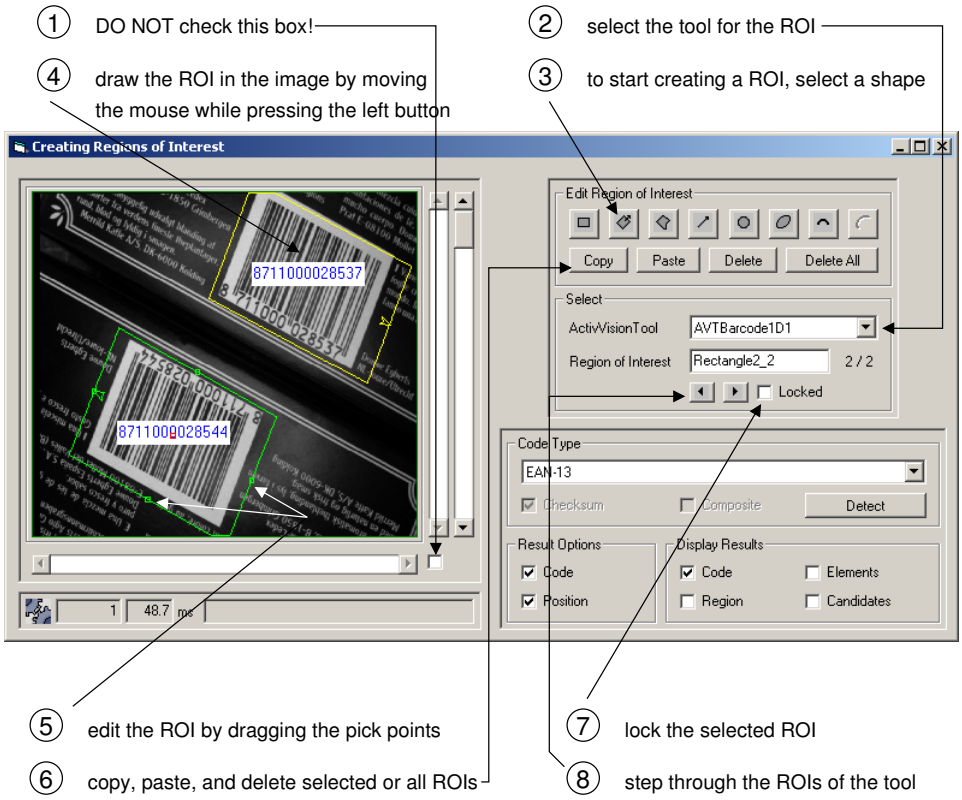


Figure 3.1: Creating and editing regions of interest.

(6) To create another ROI, repeat step (3). You can also copy, paste, and delete the selected ROI or delete all ROIs.

(7) By checking ☑ `Locked` you can lock the selected ROI and thus prevent it from accidental editing.

(8) ROIs can be selected by clicking into the vicinity of its pick points in the image. Alternatively, use the two arrow buttons ◀ ▶ to step through all the ROIs of the selected tool.

To read the bar codes, click `Detect` in `AVTBarcode1D`. To visualize results, check the boxes in the frame `Display Results`.

3.2 Displaying ROIs and Results Using

In the previous section, you already got a glimpse of image processing results. How results (and ROIs) are displayed, can be adapted via `AVTViewDisplayModes`.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project.
  Otherwise, open the project `displaymodes\view_displaymodes.vbp`.
- ☐ Execute the application and load `barcode\ean13_08`.

The following steps are visualized in figure 3.2.

① Open `AVTViewDisplayModes` by clicking on `AVTView` with the right mouse button and selecting `Display Modes` in the popup menu.

② In the frame `Display Results` of `AVTBarcode1D`, check all boxes. Using `AVTView-DisplayModes`, you can now change various display parameters, e.g., ...

③ ...the color of the currently selected ("active") ROI, of the other ROIs of the selected tool, or of the ROIs of other, not selected tools, or of ROIs that are evaluated as "not okay" by ActivDecision ...

④ ...the color of the selected ("active") pick point, ...

⑤ ...the line width of the ROIs or of the results ...

⑥ ...the LUT (look-up table) that is used to display the image ...

⑦ ...the color of the different types of results. ActivVisionTools can assign their results to two groups to allow to display them using different colors. In the case of ActivBarcode, the elements and candidates are assigned to group I, the bar code region to group II. A third color is reserved for presumed elements that could not be decoded and results that are evaluated as "not okay" by ActivDecision. Finally, you can choose the color used for highlighting results if you select them in ActivDataView or ActivDecision.

⑧ ...the color and the font of displayed text and whether it is displayed on a contrasting background ...

⑨ ...how regions are displayed and whether contours are displayed with a contrasting border.

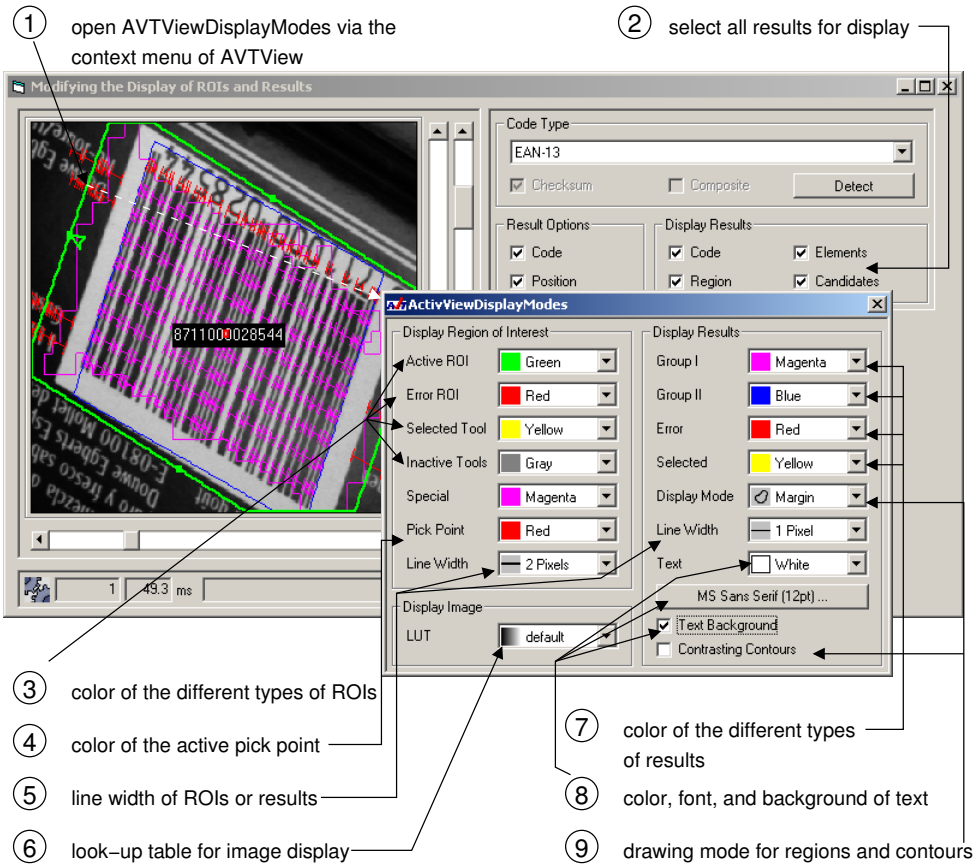


Figure 3.2: Changing display parameters.

Note that if you zoom into an image, the display of ROIs is clipped to the displayed part of the image (see e.g. figure 3.2). Thus, if you see a rectangle around the image, this means that there is an ROI encompassing the displayed image part. A very thin green rectangle around the image, in contrast, indicates that the image is the active item of your application, i.e., that interactions like keyboard input will be processed by the image.

The question how to adapt the display is also addressed in the manuals of other ActivVisionTools, with the emphasis on the special needs of the respective tool (ActivAlignment: section 4.1 on page 24, ActivGeoCalib: section 4.1 on page 24, ActivMeasure: section 4.1 on page 26, ActivBarcode: section 4.1 on page 22, ActivDataCode: section 4.1 on page 36, ActivBlobFinder: section 4.1 on page 34, ActivOCR: section 4.1 on page 58, ActivDecision: section 3.1 on page 24)

3.3 Inspecting the Status of an Application with

`AVTViewStatus` displays information about your ActivVisionTools application, e.g., the execution time of a cycle, but also warnings and error messages.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. If `AVTViewStatus` is not already on the form, please add it by double-clicking .
- ☐ Execute the application and load `barcode\code39_01`.

The following steps are visualized in figure 3.3.

① At the left side of `AVTViewStatus`, an icon indicates the current execution mode of the application. In the mode , the application does not perform any processing and waits for your interaction. If you start the continuous mode the cogwheels rotate; any interaction on your part is stored in the event queue and processed after the current cycle is finished. If the cursor gets “busy”, ActivVisionTools has started a particularly time-consuming operation, e.g., connecting to an image acquisition device. Any interaction on your part is then deferred to the end of this operation.

② In the next field, `AVTViewStatus` shows the number of processed cycles.

③ Next to it, `AVTViewStatus` shows the time needed for the last processing cycle.

④ You can change this to the number of processing cycles per second via a property: Open `AVTViewProperties` by clicking on `AVTView` with the right mouse button and selecting `Properties` in the appearing context menu. Then, select the properties of `AVTViewStatus`. Beside the mode of displaying processing time or speed, the properties also allow to modify which of the fields described above are shown.

⑤ To the right, `AVTViewStatus` displays ActivVisionTools messages. There are two types of messages: Informative messages describe what the application is doing while it is “busy” (e.g., connecting to an image acquisition device) or warn you if the image processing failed (e.g., why a bar code could not be read as in figure 3.3).

In contrast, error messages indicate errors that prevent the application from working correctly, e.g., the failure to connect to an image acquisition device.

⑥ To display the history of messages, double-click into the message field.

⑦ If `AVTViewStatus` is not added to an application, error messages are displayed in popup dialogs. If an error occurs while the application is running continuously, the dialog

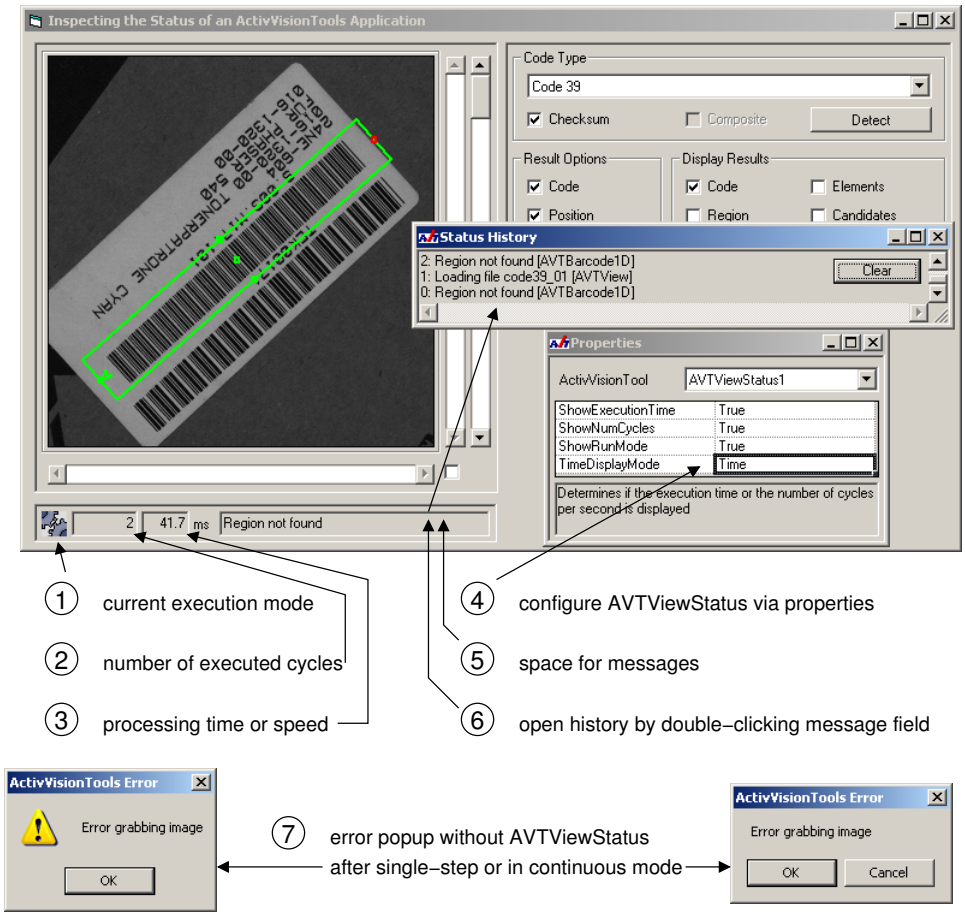


Figure 3.3: The status of an ActivVisionTools application.

offers two buttons: If you click `OK`, the application continues running. In the example, where the grab error was provoked by selecting wrong parameters for image acquisition, ActivVisionTools would again try to grab an image, resulting in the same error. In such a case, you can click `Cancel` to stop the continuous execution.

3.4 Configuring the Two Execution Modes Via and

`AVTViewExecute` allows you to switch between the two execution modes and to start an application; `AVTViewExecuteSimple` is a simplified version consisting of a single button.Using `AVTViewConfigExec`, you can customize the behavior of an ActivVisionTools application, e.g., what is displayed, for the two execution modes. This allows you, e.g., to display live images only in the *configuration mode* to setup your application, but then switch it off in the *application mode* to speed up the application.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add `AVTViewExecuteSimple` to the form by double-clicking .

  Otherwise, open the project `usermodes\view_usermodes.vbp`.

- ☐ Execute the application and load the image sequence `barcode\barcode1.seq`.

The following steps are visualized in figure 3.4 (not shown: `AVTBarcode1D`).

① Open `AVTViewExecute` and `AVTViewConfigExec` by clicking on `AVTView` with the right mouse button and selecting `Execution` and `Execution Parameters` in the popup menu.

② In `AVTViewExecute`, you can switch between the two execution modes via the combo box `Mode`.

③ To execute one cycle, press `Single`. In our example, this means that the next image of the sequence is loaded. With the other two buttons, or with the button in `AVTViewExecuteSimple`, you can start the application in a continuous mode and stop it again.

④ For each of the two execution modes, you can choose what is to be displayed by checking the corresponding boxes in `AVTViewConfigExec`. Furthermore, you can specify if images can be dragged to the image window and whether ROIs can be modified in the two modes; by default, this is disabled in the *application mode* to prevent you from accidentially moving or deleting an ROI.

⑤ By default, `AVTViewExecuteSimple` executes the application in continuous mode, but you can change its behavior to single-step execution. For this, open `AVTViewProperties` by clicking on `AVTView` with the right mouse button and selecting `Properties` in the appearing context menu. Then, select the properties of `AVTViewExecuteSimple`.

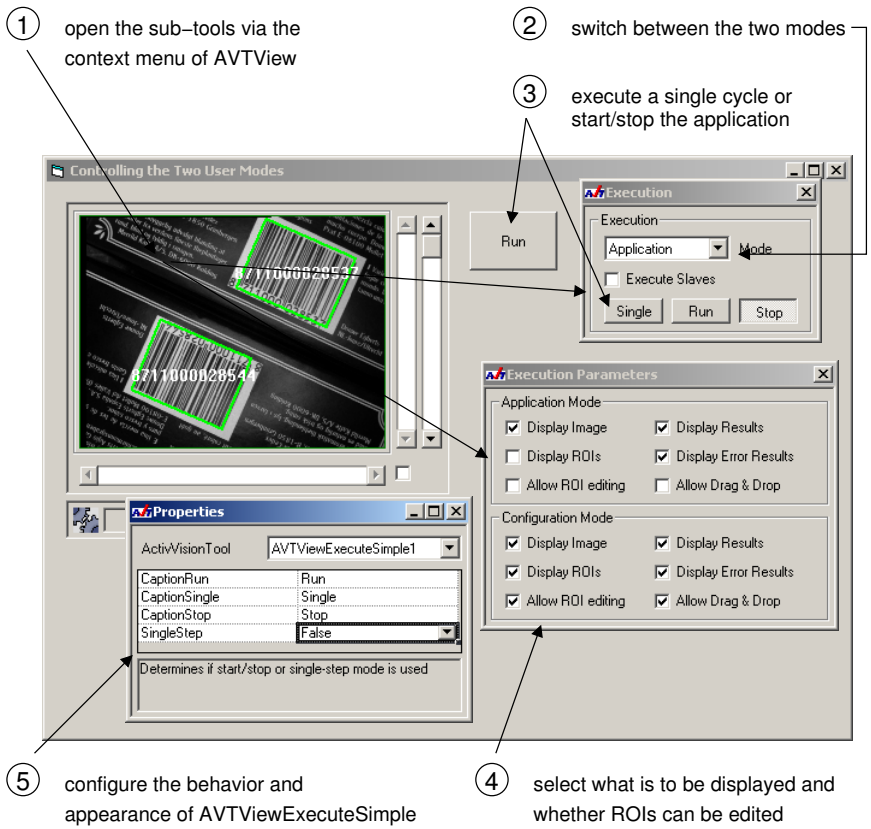


Figure 3.4: Configuring and switching between the two execution modes.

With the property `SingleStep` you can modify the behavior, with the other properties the button text.

3.5 Inspecting and Modifying Connections Between ActivVisionTools Via

ActivView is also in charge of the connections between the individual ActivVisionTools in an application. To inspect and modify the connections it provides the *support tool* `AVTView-Connections`. Note that in most applications you don't need to modify connections; exceptions are applications that use ActivGUIComponents, ActivGenericInterface, or multiple instances of ActivView or of ActivBlobFinder and ActivFeatureCalc.

Visual Basic Example

Preparation for the following example:

- ☐ We recommend to open the project `connections\view_connections.vbp`, which already contains two instances each of `AVTView` and `AVTBarcode1D`.
- ☐ Execute the application and load the images `barcode\barcode1.seq` and `barcode\code39_01`, respectively.

The following steps are visualized in figure 3.5.

① Open `AVTViewConnections` by clicking on either instance of `AVTView` with the right mouse button and selecting `Connections`. The dialog depicts the connections in form of a tree. In the example, all tools, including the second instance of `AVTView`, are connected to the first instance of `AVTView`.

② If you click on a tool in the tree, the corresponding dialog is highlighted. The tool tip displays additional information about the tool, e.g., its input and output data or its number of ROIs. If the tool is not connected to any other tool, the tool tip explains why. If a tool name appears in grey (like `AVTViewConnections`), it is not executed in the execution cycle. By clicking on a tool with the right mouse button, you can open `AVTViewProperties`.

③ You can change connections between tools by dragging them to another position in the tree. Allowed positions are marked in green. For example, the second instance of `AVT-Barcode1D` can only be connected to one of the two instances of `AVTView`. Furthermore, while you drag a tool, the cursor shape changes. If it is red (e.g., when you drag `AVT-Barcode1D2` over `AVTBarcode1D1`, the connection is not possible.

! Note that **ROIs are lost** if you connect a tool to a different instance of `AVTView`!

The lower part of figure 3.5 shows the state after connecting `AVTBarcode1D2` to `AVTView2` and creating an ROI for it. A further example of using `AVTViewConnections` can be found in the Advanced User's Guide for ActivVisionTools, section 2.2.2 on page 44.

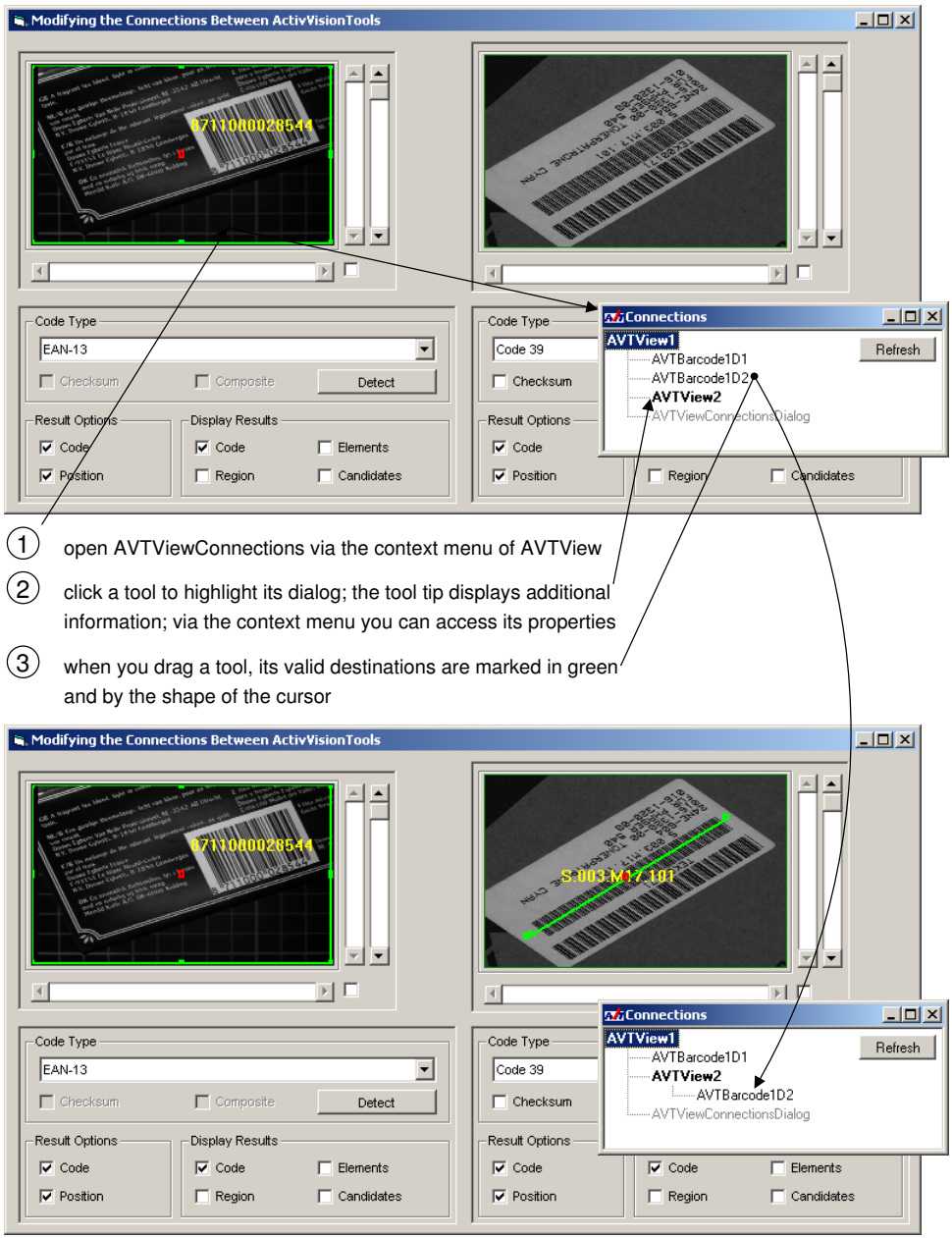


Figure 3.5: Modifying connections between ActivVisionTools .

Chapter 4

Tips & Tricks

This chapter contains additional information that helps you to get optimal results when using ActivView. Section 4.1 shows how to use the *inspection tools* ActivHistogram and ActivLineProfile to analyze image quality. The chapter ends with a list of problems you might encounter when accessing your image acquisition device from ActivView together with tips how to solve them.

4.1 Analyzing Image Quality 40
4.1.1 A First Look at ActivHistogram and ActivLineProfile 40
4.1.2 Analyzing Saturation 42
4.1.3 Analyzing Sharpness 44
4.1.4 Other Aspects of Image Quality 46
4.2 Troubleshooting 48

4.1 Analyzing Image Quality Using and

When developing machine vision applications it is important to keep in mind that high quality results can only be extracted from high quality images. This section shows how to analyze and enhance the quality of acquired images using the *inspection tools* ActivHistogram and ActivLineProfile.

4.1.1 A First Look at ActivHistogram and ActivLineProfile

ActivHistogram calculates the gray value distribution within a region of interest, i.e., how often individual gray values occur within the region, while ActivLineProfile analyzes the gray value profile, i.e., the sequence of gray values along an ROI. In this section, we give a quick tour of the two tools. How to "read" the histogram and the profile, i.e., how to get information to optimize your machine vision setup with regard to image quality, is the topic of the following sections.

Visual Basic Example

Preparation for the following example:

- ☐ We recommend to open the project `imagequality\view_imagequality.vbp`.
- ☐ Execute the application and load the image `misc\saturation_01`.

The following steps are visualized in figure 4.1.

1. First, create an ROI for `AVTHistogram1` using `AVTViewROI` as described in section 3.1 on page 28. Automatically, the histogram of this region is displayed, with arrow heads marking the minimum, maximum, and peak gray value. Below the histogram, the corresponding gray values and further statistical information is displayed.

   If you don't create an ROI for `AVTHistogram`, it uses the full image.

2. With a slider you can zoom into the lower part of the histogram.

3. Two vertical lines form a lower and an upper threshold. The default values for the thresholds are 0 and 255. You can move them by dragging, i.e., by keeping the left mouse button pressed.

4. In the combo box `Display` you can select what is to be displayed in `AVTView`. In figure 4.1, the pixels `'Inside'` the two thresholds are displayed.

5. To inspect a gray value profile, you create the line (or arc) of inspection using `AVTViewROI` as described in section 3.1 on page 28. The gray value profile along that line is displayed automatically. Note that in a rectangular ROI the pixel values are averaged perpendicularly to the main axis of the rectangle.

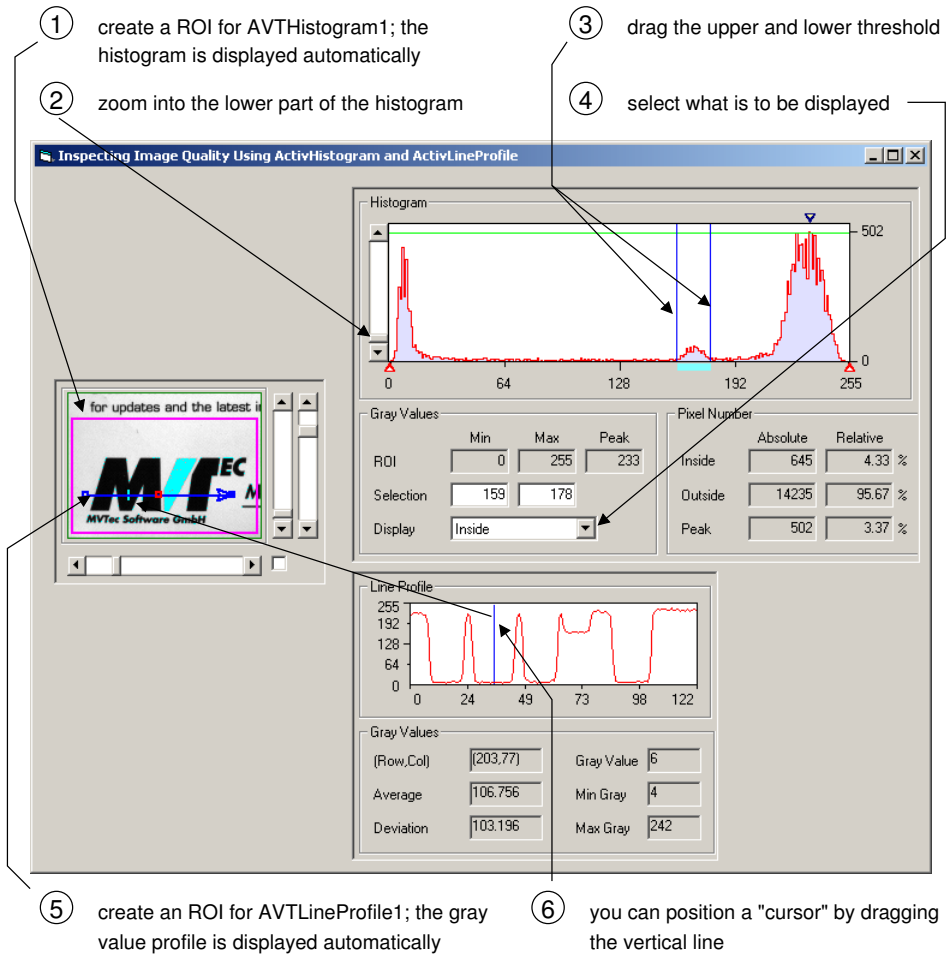


Figure 4.1: Inspecting the image using ActivHistogram and ActivLineProfile .

6 For an easier inspection, ActivLineProfile provides a “cursor” which can be moved along the line. The position of this cursor is marked by vertical lines in `AVTLineProfile` and on the corresponding ROI in the image. The position itself is displayed in the text box `(Row,Col)`, the gray value in the text box `Gray Value`. You can move the cursor by dragging the line in `AVTLineProfile` with the left mouse button.

4.1.2 Analyzing Saturation Using

Within a vision system the intensity of light sensed by the camera is mapped to the available gray values (e.g., 256 in an 8-bit image). Because the range of gray values has two boundaries (e.g., 0 and 255), the mapping is not continuous, i.e., all intensities above a certain limit are assigned the value 255, and those below a second limit are assigned to 0. If, e.g. the illumination is too bright or the aperture of the lens too wide, the intensity of parts of the image may lie above the upper saturation limit (i.e., 255 in an 8-bit image). In this case, all these parts appear with the value 255 in the image; thus, they cannot be distinguished by their gray value even if their intensity differs in the real world. This effect is called *saturation*; it can be analyzed easily using `AVTHistogram`.

Visual Basic Example

Preparation for the following example:

- ☐ Please continue with the previous example.
- ☐ Execute the application and load the image sequence `misc\saturation1.seq`. Place an ROI around the logo as shown in figure 4.2.

The following steps are visualized in figure 4.2 (not shown: `AVTLineProfile`).

① The well-balanced image shows three distinctive histogram peaks for the bright background, the dark characters, and the lighter character (marked with the two threshold bars).

② The second and the third image of the sequence were acquired with a too wide aperture; both are saturated at high gray levels (“over-saturated”). The peak corresponding to the (bright) background has “melted” into the right boundary, the peak value is 255; ActivHistogram indicates this state by highlighting the two markers for the maximum and the peak value in red.

As a consequence, the peaks corresponding to the characters have shrunk. Even worse, the area of the characters, measured by the number of pixels, has shrunk as well; this leads to inaccuracies when inspecting those regions, e.g., using ActivMeasure or ActivFeatureCalc.

③ The fourth and the fifth image of the sequence were acquired with a narrow aperture; both are saturated at low gray levels (“under-saturated”). Now, the peak corresponding to the dark characters has melted into the left boundary, the peak value is 0; now, the two markers for the minimum and the peak value are highlighted in red.

As a consequence, the peaks corresponding to the light character to the background have shrunk together with their area. Furthermore, the range of gray values is not used fully anymore; the maximum gray value appearing in the image is 206.

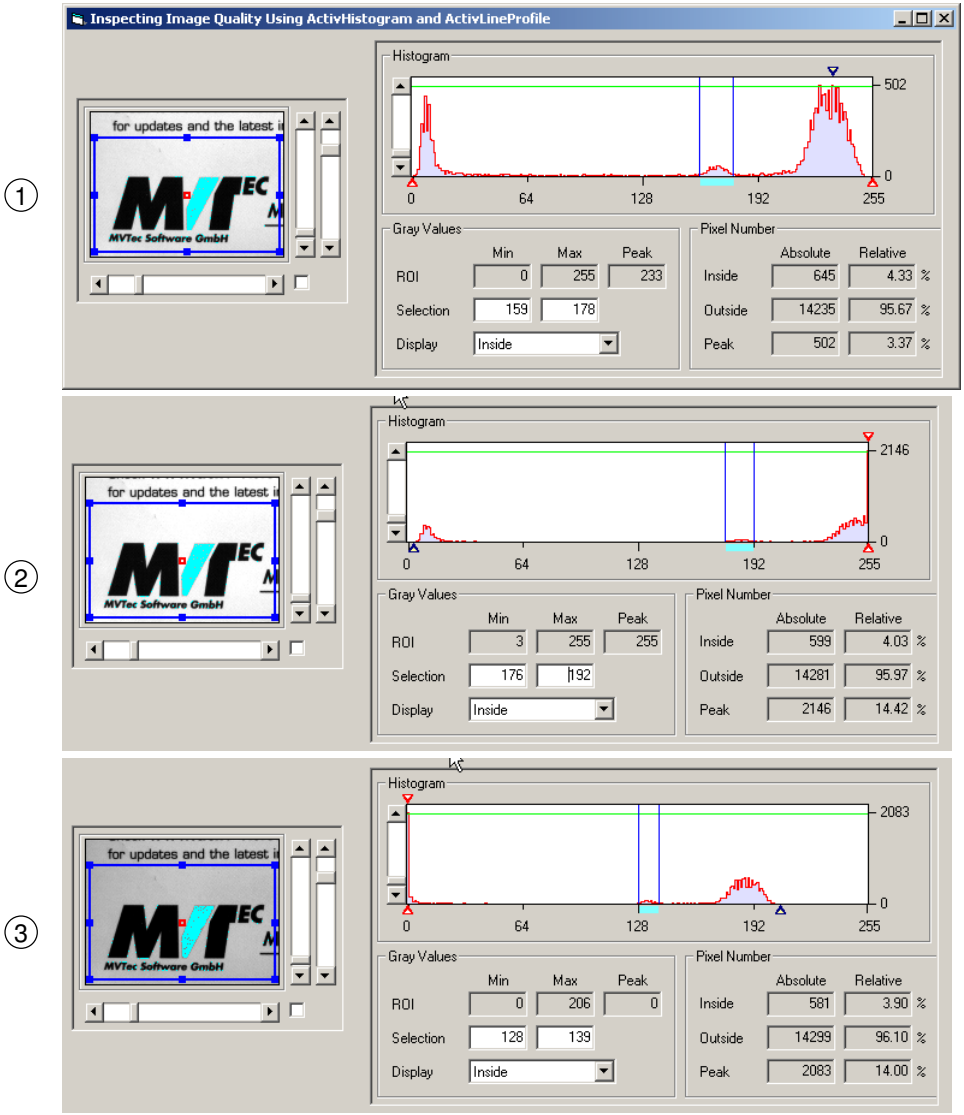


Figure 4.2: Histograms of well-balanced, over-saturated, and under-saturated images.

If you are grabbing from a camera, please experiment with different apertures and illuminations. If your camera provides features like an automatic gain or automatic iris, please switch them off in order to get repeatable settings. Furthermore, switch off any on-camera gamma correction, because it reduces the accuracies when using ActivMeasure or ActivFeatureCalc.

4.1.3 Analyzing Sharpness Using

The sharpness of an image can be estimated from looking at image edges, i.e., discontinuities in gray values. In a sharp image, the gray values change abruptly at an edge. This can be analyzed easily using `AVTLineProfile`.

Visual Basic Example

Preparation for the following example:

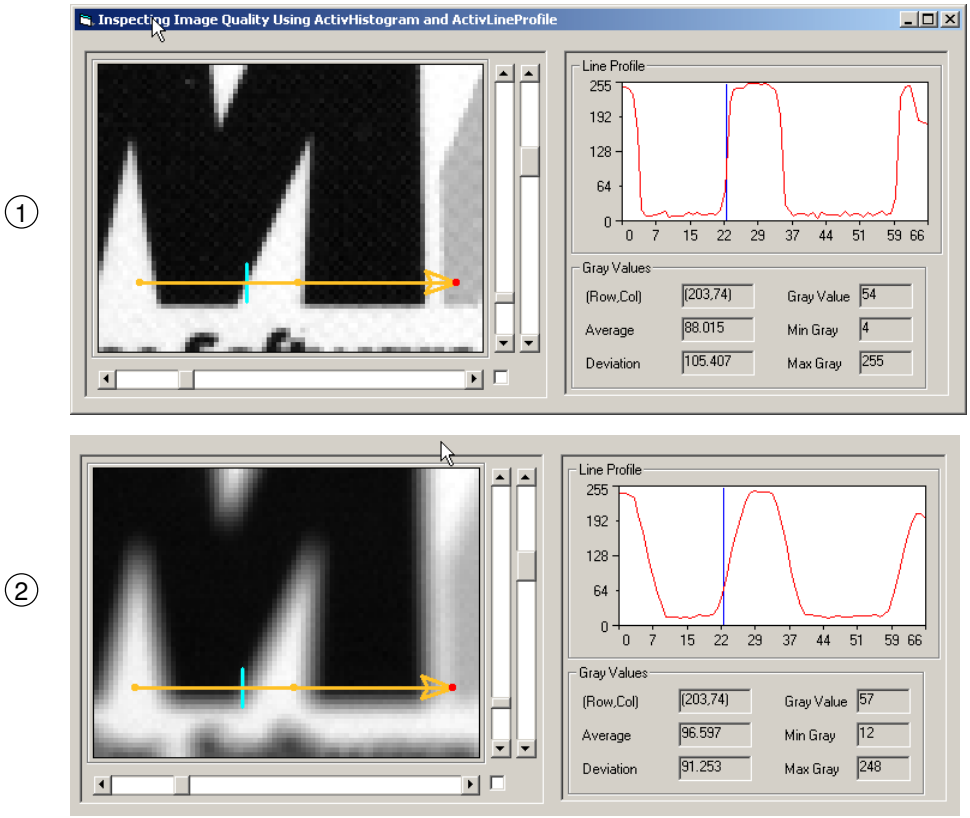
- ☐ Please continue with the previous example.
- ☐ Execute the application and load the image sequence `misc\sharpness1.seq`. Place an ROI across the characters as shown in figure 4.3.

The following steps are visualized in figure 4.3 (not shown: `AVTHistogram`).

① In the sharp image, gray values change abruptly at edges; the gray value profile shows steep transitions between the gray values corresponding to the characters and the background, respectively.

② The second image of the sequence was acquired after defocusing. In the gray value profile, the gray values now change more gradually between the characters and the background.

If you are grabbing from a camera, please experiment with the focus.



① ②

Figure 4.3: Gray value profile of sharp and defocused image.

4.1.4 Other Aspects of Image Quality

In the previous sections we showed how to use the information provided by ActivHistogram and ActivLineProfile to optimize the setup of your vision system. In this section, we take a brief look at other aspects of image quality which show up in the histogram.

Visual Basic Example

Preparation for the following example:

- ☐ Please continue with the project of the previous examples.
- ☐ Execute the application.

The following steps are visualized in figure 4.4 (not shown: `AVTLineProfile`).

① Load the image `misc\saturation_01_scaled` and place an ROI across the logo as shown in figure 4.4. The histogram shows regular gaps, i.e., at regular intervals gray values do not appear in the image. If you see such a histogram this usually means that the gray values of an image have been scaled after the image has been digitized.

In fact, the image `misc\saturation_01_scaled` has been created from the image `misc\saturation_01` (load this image or see figure 4.1 on page 41) by downscaling the gray values with the factor 0.25 and then back again with the factor 4; the two images seem to be identical, but the histogram clearly shows that information has been lost by the scaling.

② Now load the image sequence `misc\noise1.seq`, place an ROI over the light character (see figure 4.4), and click `Run`. Especially if you zoom further into the image you can see that the pixel values vary slightly due to noise.

You can visualize noise even better using `AVTHistogram`: Place the two threshold bars around the peak and select `'Scale Image'` in the combo box `Display`. Now, the gray values in the ROI are transformed in such a way that the interval selected by the two thresholds is mapped to the maximum interval, i.e. $[0..255]$. Thus, the variations of the gray values appear more prominently. Note that `AVTHistogram` just displays the scaled gray values overlaid on the image, it does not change the gray values themselves.

If you are grabbing from a camera, take a look at the noise it is producing. Furthermore, you can test whether your vision system produces the so-called dark current noise: Make sure that no light enters the camera, e.g., by closing the aperture; now, all pixels should have the gray value 0. If this is not the case, you can visualize the noise by placing the threshold bars around the peak and selecting `'Scale Image'`.

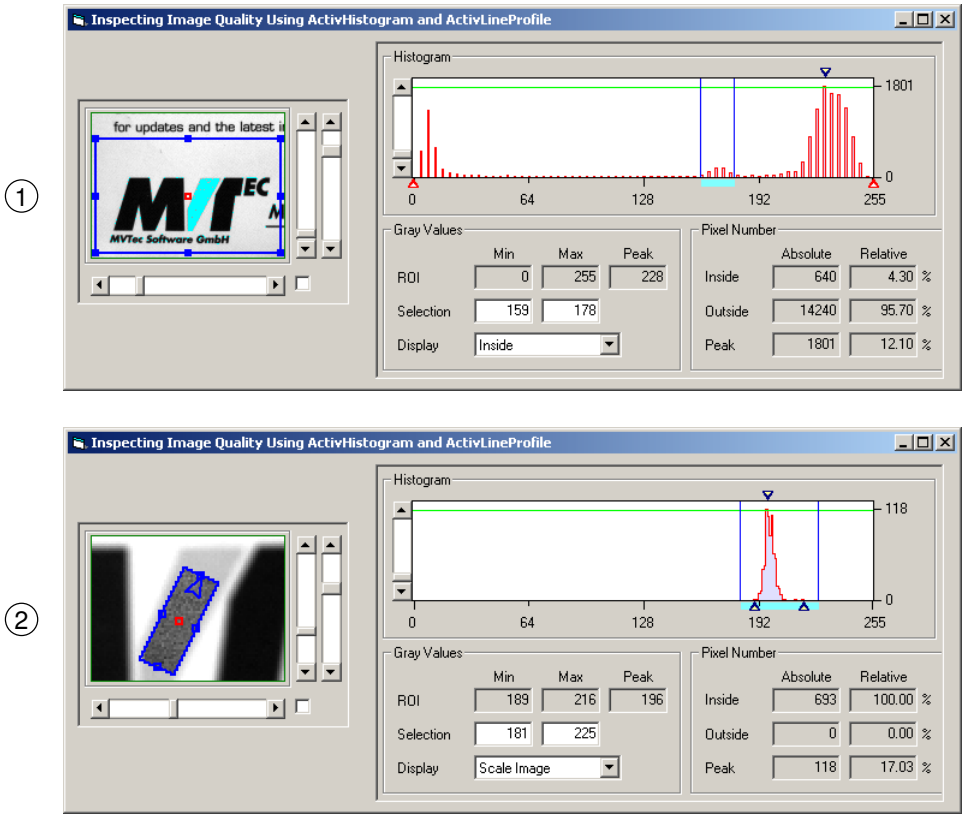


Figure 4.4: Other aspects of image quality: ① scaled gray values; ② noise.

4.2 Troubleshooting

In this section you find information about how to solve problems regarding image acquisition. If the information given does not solve your problem, please have a look at the ActivVisionTools FAQ which can be found under `http://www.activ-vision-tools.com`. If your question is still not answered, send a problem report by email to your local distributor, not MVTec directly. See the Getting Started with ActivVisionTools, section 4.3.3 on page 48, for more information.

? The image acquisition device does not appear in the combo box after clicking `Detect`

Please check whether you correctly installed the image acquisition device. As a first step, use the test program most manufacturers provide with their device to check whether images are actually grabbed. If it does not, maybe the corresponding device driver was not installed successfully, or the driver is not started (check the Windows Device Manager; the driver should be started automatically).

If the driver is running, check whether the environment variable `PATH` contains the directory where the DLLs of the SDK of the image acquisition devices reside.

Finally, check whether the directory `%HALCONROOT%\bin\x86-win32` contains the HALCON image acquisition interface for your image acquisition device, i.e., the DLL `HFGxyz.dll` with `xyz` being the name of your device interface (`%HALCONROOT%` is the folder you selected during the installation of HALCON, e.g., `C:\Program Files\MVTec\Halcon`).

If everything seems to be okay, please contact your local distributor.

? Connecting to an IDS or DFG-LC frame grabber fails and an error popup window appears

Some parameter selections in the frame `Device` may cause an internal error in the frame grabber SDK which results in the error popup.

? MATRIX Vision frame grabbers report not enough memory

If you get an error in ActivVisionTools claiming that not enough memory is available whereas MATRIX Vision's test program grabs images continuously, check whether you allocated enough memory for two image buffers in MATRIX Vision's configuration program `MVconfig`.

? The grabbed image has a black part at the bottom

If you selected a PAL signal, try NTSC instead. If you selected the automatic signal analysis (`'auto'`) also try NTSC.

? **The grabbed image is distorted completely**

Maybe you selected NTSC instead of PAL, or `'Gray 16'` instead of `'Color 24'`.

? **The grabbed image appears to be stretched**

Resize the display to fit the size of the image as described in section 2.3 on page 20.

? **The external trigger does not work within Visual Basic**

If you are using ActivVisionTools within the Visual Basic environment, in some cases the external trigger mechanism does not work. To test whether this is caused by the Visual Basic environment, create a stand-alone application (see the manual Getting Started with ActivVisionTools, section 3.5 on page 38); here, the external trigger should work.

A general note for those who switch between image acquisition devices of different manufacturers: We recommend to keep only the currently used boards in the computer and to uninstall the software of unused devices (or at least stop the corresponding driver). Otherwise, you might get unexpected errors, as some image acquisition devices interfere with each other.

Appendix A

Accessing Your Image Acquisition Device

This appendix contains more detailed information about the supported image acquisition devices. Section A.5 on page 57 describes how to set additional parameters of your image acquisition device using `AVTViewFGParameter`.

A.1 Supported Image Acquisition Devices 53
A.2 More Information About the Supported Image Acquisition Devices 53
A.3 Detection of Image Acquisition Devices 55
A.4 Modifying the List of Available Image Acquisition Devices 56
A.5 Setting Additional Parameters 57
A.5.1 Which Parameters Can Be Set for an Image Acquisition Device ? . . . 57
A.5.2 How to Set Parameters Interactively 58
A.5.3 When Are Parameters Set? 62
A.5.4 How to Set and Read Parameters Via the Programming Interface 62
A.6 How to Access an Unsupported Image Acquisition Device 64
A.6.1 Accessing Your Own HALCON Image Acquisition Interface 64

A.6.2 Integrating Images Directly 67

A.1 Supported Image Acquisition Devices

Table A.1 lists the currently supported image acquisition devices. The column *Name* corresponds to the entry which appears in the the combo box `Name` in `AVTViewFG`. In the Start Menu entry for ActivVisionTools, you can find an overview page with links to the HTML documentation of the interfaces.

If the image acquisition device you want to use is not in the list, it is nevertheless possible to use it together with ActivVisionTools. Please refer to section A.6 on page 64 for more details.

Please refer to `http://www.mvtec.com/halcon/image-acquisition/` for the latest information about the image acquisition devices supported by ActivVisionTools. Note that if you download a new interface you must adapt the list of image acquisition devices as described in section A.4 on page 56.

A.2 More Information About the Supported Image Acquisition Devices

Section 2.2 on page 18 showed how to connect to your image acquisition device by setting the corresponding parameters in `AVTViewFG`. However, the values available for each device may differ widely. They are documented in detail in the HTML page for each HALCON image acquisition interface – but in the nomenclature of the corresponding HALCON operator, `open_framegrabber`. You can open these HTML pages via the Start Menu entry for ActivVisionTools.

Table A.2 on page 55 helps you to translate between these two worlds by showing which parameter or value of the HALCON operator `open_framegrabber` corresponds to the elements or values of `AVTViewFG`.

The following list contains additional information for some interfaces that is important when using them with ActivVisionTools:

`DT315x`
: If you select `'volatile'` in `CamType`, images are not stored in memory but overwritten continuously, using two buffers. This speeds up image grabbing significantly. However, **volatile grabbing must not be used when working with ActivGeoCalib!** Furthermore, volatile grabbing is not supported with color images.

`INSPECTA`, `INSPECTA5`
: You select a camera configuration in two steps: First, load a configuration file via the file selector box (`Choose File`). Then, type the name of the section within this file which corresponds to your camera into the lower text box.

Company, Type / Family	Name
Universal interface for IIDC 1394-compliant IEEE 1394a/b (FireWire) cameras	1394IIDC
Universal DirectFile interface	DirectFile
Universal DirectShow interface	DirectShow
Universal interface for GigE Vision-compliant cameras	GigEVision
Universal TWAIN interface	TWAIN
Virtual acquisition interface (File interface)	File
ABS UK1000/3000 cameras	ABS
Basler eXcite smart camera	eXcite
Basler GigE Vision and IEEE 1394 cameras via pylon library	pylon
Baumer FWX/FWXC cameras	BaumerFCAM
BitFlow Raven, RoadRunner, R3, R64, and R64e	BitFlow
Daheng USB 2.0 cameras	DahengCAM
DALSA boards and Genie cameras via Sapera LT library	SaperaLT
Data Translation DT3152, DT3153, and DT3155	DT315x
Data Translation DT3162	DT3162
dPict Imaging Aexeon and Nexeon	dPict
Eltec p3i2	p3i2
Eltec p3i4	p3i4
Euresys DOMINO, GRABLINK, PICOLO, and QUICKPACK	MultiCam
IDS FALCON and EAGLE	FALCON
IDS uEye USB 2.0 cameras	uEye
Imagenation PX510, PX610, and PX610A	PX
Imagenation PXC200 and PXC200A	PXC
Imagenation PXD1000	PXD
Imagenation PXR800	PXR
Integral Technologies FlashBus MV	FlashBusMV
Integral Technologies FlashBus MX	FlashBusMX
Leutron PicPort, PicPortPro, PicProdigy, and PicSight	Leutron
LinX Ginga++	Ginga++
LinX Ginga digital	GingaDG
LinX GINGA digital-CL1e/CL2e/CL4e and GINGA++M1e/M4e	LinX
Lumenera USB 2.0 cameras	LuCam
MATRIX VISION mvBlueFOX, mvDELTA, mvGAMMA, mvSIGMA, and mvTITAN	MatrixVisionAcquire
Matrox Meteor-II, Helios, Solios, and Odyssey	MILLite
Mikrotron INSPECTA-3 and INSPECTA-4	INSPECTA
Mikrotron INSPECTA-5	INSPECTA5
Opteon imaging systems	Opteon
Pleora iPORT IP Engine	iPORT
SICK|IVP Ranger C	RangerC
SICK|IVP Ranger E	RangerE
Silicon Software microEnable III	mEnableIII
Silicon Software microEnable IV	mEnableIV
Silicon Software microEnable boards via VisualApplets	mEnableVisualApplets
Sony XCI-SX1 and XCI-V3 smart camera	SonyXCI
Tattile TAG cameras	TAG
VRmagic USB 2.0 cameras	VRmUsbCam
The Imaging Source DFG/LC1, DFG/LC2, DFG/LC4, DFG/MC2, and DFG/MC4	DFG-LC

Table A.1: Currently supported image acquisition devices.

AVTViewFG	open_framegrabber
frame Format: CamType, Choose File, Camera, Mode, etc.	CameraType
Color	ColorSpace
Board, Source	Device
Port	Port
'default'	-1, default

Table A.2: Translation between parameter names in AVTViewFG and the documentation of the HALCON image acquisition interfaces.

Leutron
: If your configuration is not in the list of the combo box Board, you may type it directly into the text field of the combo box. For example, the entry for a configuration consisting of 2 (physical) boards with one camera connected to the first and two cameras connected to the second board would be 'Picport Stereo H4-D:1:2:2'.

MultiCam
: The frame grabbers supported by this interface use two auxiliary files, called *mpf* and *camera file* to adapt the grabbing to various cameras. To select a camera file, click on the button Choose File to open a file selection box. The selected file (without suffix .cam) is displayed in the upper combo box. The mpf file can be selected in the lower combo box. For some board types, you can also specify the board topology by inserting the corresponding value (e.g., '1_1') before the name of the mpf file, separated by a colon (':').

A.3 Detection of Image Acquisition Devices

Upon starting, ActivView inspects the host computer for image acquisition interfaces. In a first step, ActivView queries HALCON for a list of all supported image acquisition interfaces. Furthermore, it scans the dynamic libraries for additional image acquisition interfaces, which are then also added to the list in the combo box Name.

Of course, the presence of the DLLs does not necessarily mean that actual image acquisition devices are available for the corresponding interfaces. Therefore, when clicking Detect in AVTViewFG, ActivView checks the presence of the devices by trying to communicate with them; however, not all image acquisition device SDKs allow this or create errors when trying to access a not-existing device. Nevertheless, if an image acquisition device is detected this way, it becomes the default selection in the combo box Name. If no image acquisition device could be detected, 'File' becomes the default.

If you override the default selection, your choice will be saved in the description file (DSC file); thus, when starting an application the next time, this choice will be restored.

Note that some image acquisition devices of different manufacturers cannot be distinguished by the automatic detection mechanism. Therefore, the list might contain devices that you never installed on your system.

A.4 Modifying the List of Available Image Acquisition Devices

By default, ActivView asks HALCON which image acquisition interfaces are supported and scans HALCON's installation directory for additional image acquisition interfaces as described in section A.3 on page 55. Therefore, if you have downloaded and installed a new HALCON image acquisition interface from MVTec's web server, it will be listed automatically.

You can override the list of image acquisition interfaces by modifying the file `AVTView.nls` (not to be confused with `ActivView.nls`!), which resides in the subdirectory `nls` of the folder where you installed ActivVisionTools. In this file, add the line

```
AVTView.FGSelection                A list-of-image-acquisition-devices
```

with `list-of-image-acquisition-device` containing those devices you want to appear in the combo box, separated by a semicolon. If the line already exists in the file, you may modify it accordingly. Note that **if ActivView finds this line, it will only check the names appearing in the list** when detecting image acquisition interfaces (initiated by clicking `Detect` in `AVTViewFG`).

Note that it is not necessary to include `File` in the list to be able to load images from disk; this entry appears automatically in the combo box.

There are three situations in which you need to modify the list of available image acquisition devices:

1. The automatically detected list contains image acquisition devices which you do not want to appear in the combo box.
2. You want to use your own HALCON image acquisition interface within ActivVisionTools (see section A.6 on page 64).
3. In rare configurations, incompatible driver DLLs from older installations of image acquisition devices cause problems for the auto detection of image acquisition devices. As a result, ActivViewer may crash when clicking `Detect` in `AVTViewFG`.

A.5 Setting Additional Parameters

AVTViewFG provides the basic functionality for acquiring images which is common to all image acquisition devices. However, many image acquisition devices provide additional functionality, e.g., they might allow to modify the timeout after which to abort a pending grab or to grab only parts of the image (cropping). Moreover, some image acquisition devices allow to influence camera settings, e.g., modify the brightness of images or switch the camera to asynchronous reset mode.

This functionality can be accessed interactively using the *support tool* AVTViewFGParameter, which can be opened via AVTViewFG. This method is described in the following sections, while section A.5.4 on page 62 describes how to achieve this via the programming interface.

A.5.1 Which Parameters Can Be Set for an Image Acquisition Device ?

Which parameters can be influenced depends on your image acquisition device and the corresponding HALCON image acquisition interface (see section 1.3.1 on page 8). The image acquisition interfaces query the available parameters dynamically.

Detailed information about each parameter can be obtained from the documentation of the HALCON image acquisition interfaces, which can be found via the Start Menu entry of ActivVisionTools or under `http://www.halcon.com/image-acquisition`. For each interface, a separate HTML page exists. Scroll down this page to the section *Parameters for set_framegrabber_param*, where all available parameters are listed and described.

A.5.2 How to Set Parameters Interactively Using

Visual Basic Example

Preparation for the following example:

- ☐ You can use an arbitrary ActivVisionTools project. Execute the application, open `AVTViewFG` via the context menu of `AVTView`, and connect to your image acquisition device.

The following steps are visualized in figure A.1.

1. To open `AVTViewFGParameter`, click the button `More Parameters` in `AVTViewFG`.
2. If you have already connected to your image acquisition device (signaled by the check box ☑ `Connected`, which can be used to disconnect and connect again), information about the underlying HALCON image acquisition interface (see section 1.3.1 on page 8) is displayed in the text box `Interface DLL`.
3. For certain image acquisition interfaces the parameters may be classified by `Category` and by `Visibility`. Selecting different categories from these combo boxes confines the selection of parameters displayed in the frame `Acquisition Interface Parameters`. If no classification is available, the combo boxes are grayed out, and all available parameters are displayed.
4. The parameters can be edited in the frame `Acquisition Interface Parameters`. Depending on the parameter type additional elements are added, e.g., a slider for integer parameters or an editable combo box for strings. For some parameters, an additional check box appears via which you can select the value `auto`. Action parameters (those commencing with do) appear as buttons.

We continue with the example on the next double page.

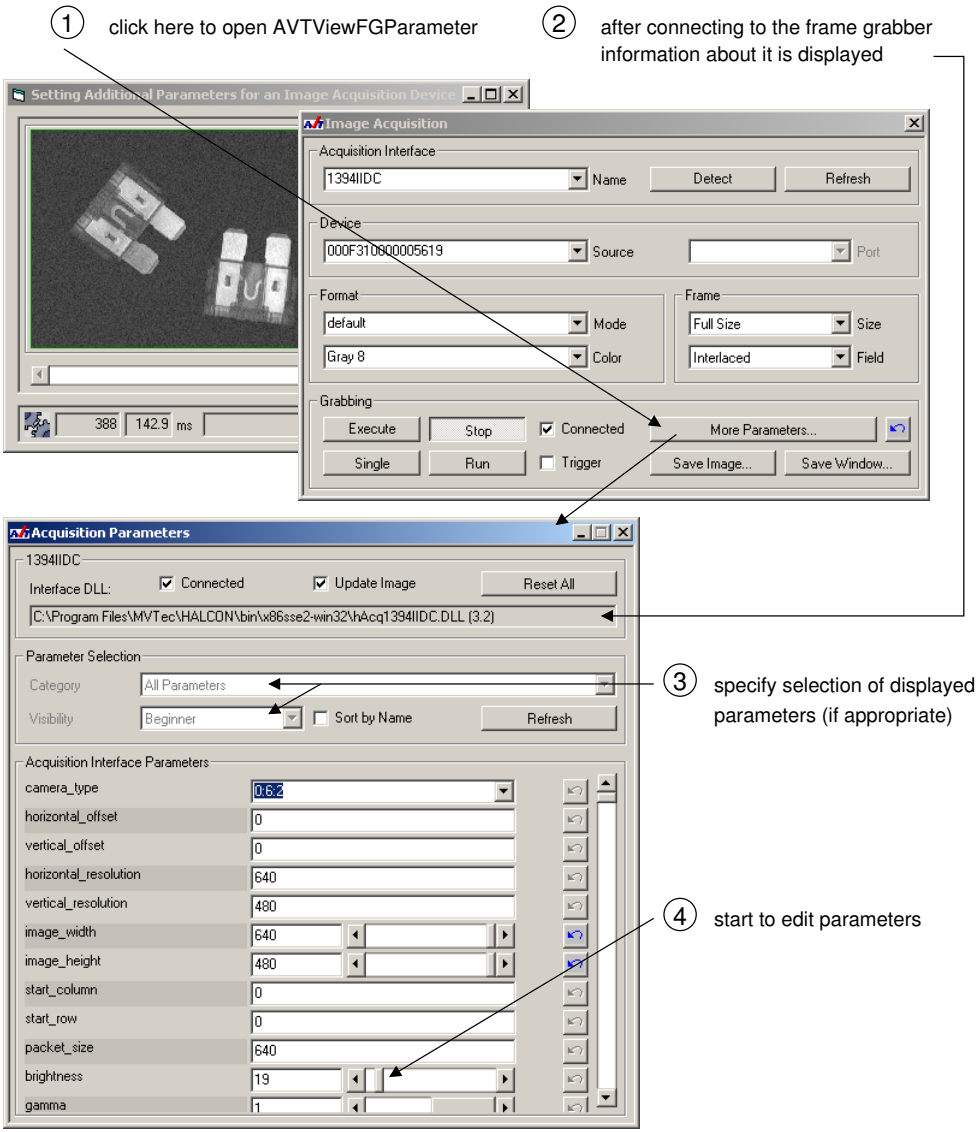


Figure A.1: Setting parameters using `AVTViewFGParameter` .

! Before we show how to modify parameters, some words of warning: Please keep in mind that **parameters should be modified with care, after reading the online documentation!** Otherwise, you might cause your ActivVisionTools application to hang or crash.

! Another warning regards the so-called *volatile grabbing* which is offered by some image acquisition devices. If you select this mode, images are not stored in memory but overwritten continuously, using two buffers. This speeds up image grabbing significantly. However, **volatile grabbing must not be used when working with ActivGeoCalib!**

We continue with the example of the last double page. The following steps are visualized in figure A.2.

(5) If the box ☑ `Update Image` is checked, an image is grabbed when a parameter is changed. Therefore, the result can be judged immediately. You can uncheck this option if you want to make a series of changes or if parameters must be set together, and turn it on again to check the result.

(6) Individual parameters can be reset to their default values by clicking the corresponding reset button. The default values set by the HALCON image acquisition interface or the SDK of the image acquisition device are used, which can be found in the HTML documentation of HALCON image acquisition interfaces (`http://www.halcon.com/image-acquisition`).

To reset all parameters at once, click the button `Reset All`.

(7) You can update the parameter list (and value ranges) by clicking the button `Refresh`. This is useful when a parameter change causes different value ranges for other parameters, e.g., when changing the camera type.

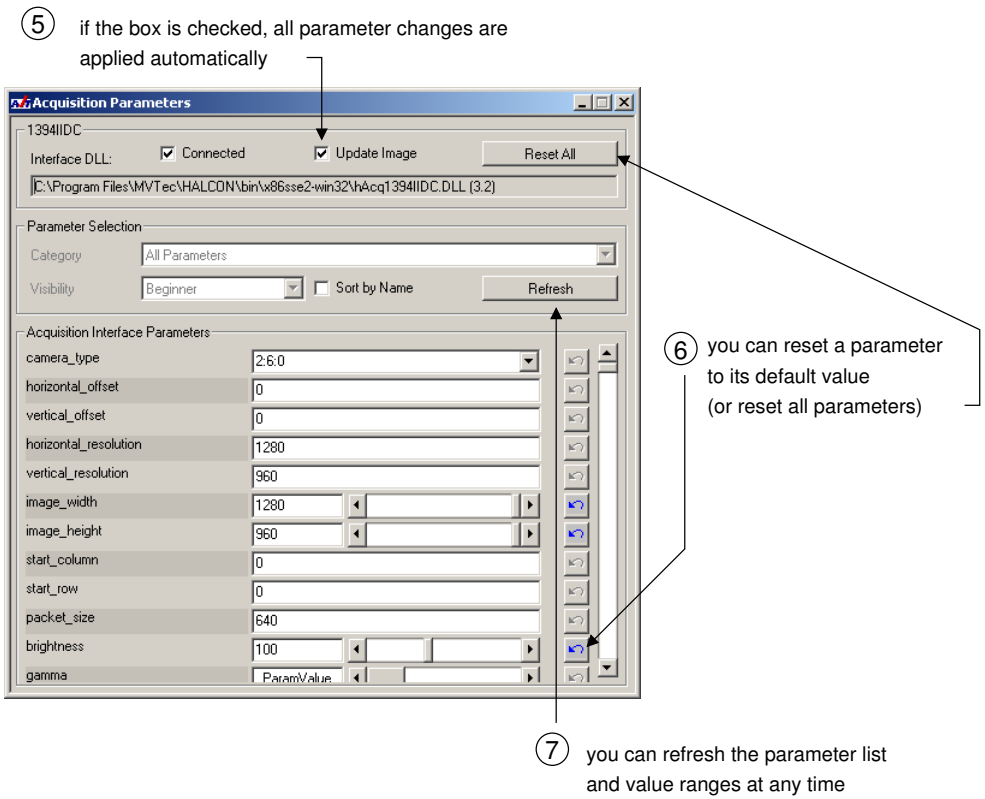


Figure A.2: Setting parameters using `AVTViewFGParameter` (continued).

A.5.3 When Are Parameters Set?

As described above, parameters are set whenever you change their value (if ☑ `Update Image` is checked) or by explicitly grabbing an image. The changed parameters are furthermore saved in the project's *description file* (DSC file, see the manual Getting Started with ActivVisionTools, section 1.2.4 on page 9, for more information) when you stop the application or select a different image acquisition device or `'File'` in `AVTViewFG`'s combo box `Name`. When you start the application (or switch back to your image acquisition device) the parameters are read from this file; they are set when you connect to the image acquisition device.

If you need to modify parameters at arbitrary moments in your application, you can do so via the programming interface as described in the following section.

A.5.4 How to Set and Read Parameters Via the Programming Interface

You can set and read parameters for image acquisition devices also via the ActivVisionTools programming interface, using `AVTView`'s methods `SetFGParameterValue` and `GetFGParameterValue`. This allows you, e.g., to set and read parameters at arbitrary moments in your application. Applications where this may be necessary include acquiring images from multiple cameras via port switching or accessing digital I/O lines provided by a frame grabber.

Figure A.3 shows an application that allows to switch between cameras (ports) via a text box and a button and that displays the current port in a second text box. The code to realize this behavior is discussed briefly below. A detailed example for port switching can be found in the Advanced User's Guide for ActivVisionTools, section 5.2.1 on page 147.

① **Setting parameters**

Parameters are set by specifying their name and the desired value. The following example code reads a value from a text box and uses it to switch to another camera connected to the specified port:

```
Dim iPort As Integer

iPort = TextPort.Text
Call AVTView1.SetFGParameterValue("port", iPort)
```

Note that when you use the programming interface of ActivVisionTools, you leave the safe world of the graphical user interfaces where all input is checked for validity automatically. Therefore, we recommend to include error handling code, e.g. as follows:

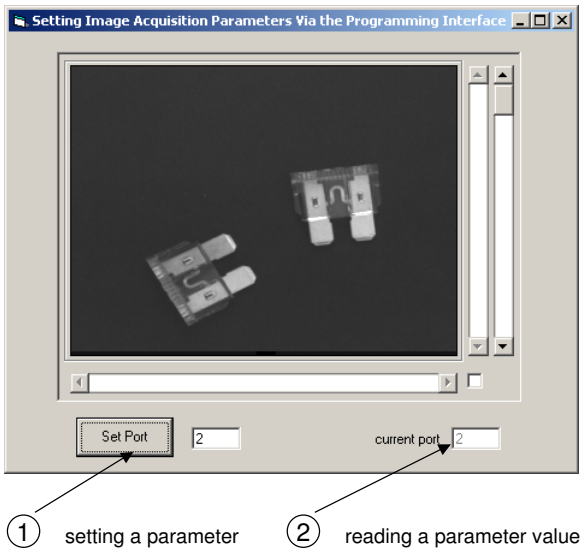


Figure A.3: Setting and reading the port via the programming interface.

```
On Error GoTo ErrorHandler

  ' code accessing the programming interface

  Exit Sub

ErrorHandler:
  Dim sTitle As String
  If Left(Err.Source, 11) = "ActivVTools" Then
    sTitle = "ActivVisionTools Error"
  Else
    sTitle = "Runtime Error " & CStr(Err.Number)
  End If
  Call MsgBox(Err.Description, vbExclamation, sTitle)

End Sub
```

② **Reading parameters**

To read a parameter's value you simply specify its name. The following code uses this to read the currently selected port, which is then displayed in a text box:

```
Dim iCurrentPort As Integer

iCurrentPort = AVTView1.GetFGParameterValue("port")
TextCurrentPort.Text = iCurrentPort
```

A.6 How to Access an Unsupported Image Acquisition Device

If you want to use an image acquisition device which is currently not supported by ActivVisionTools, there are two principal ways: If you are a HALCON user, you might have already created a corresponding HALCON image acquisition interface; section A.6.1 describes how to access this interface from ActivView. Besides this, you may feed images directly into ActivView; this is explained in section A.6.2 on page 67.

A.6.1 Accessing Your Own HALCON Image Acquisition Interface

The task of accessing your own HALCON image acquisition interface includes three steps: First, the interface must be made "visible", i.e., it should appear in the combo box `Name`. Secondly, `AVTViewFG` must adapt its content when the image acquisition device is selected. And thirdly, you specify which additional parameters can be accessed in `AVTViewFGParameter`.

① **Making the interface available**

The simplest way to select your HALCON image acquisition interface is to type its name in the combo box `Name`. This setting is then automatically stored in the DSC file of your application.

Instead of typing the name in each application, you can also add it to list of combo box items by modifying the file `AVTView.nls`, which resides in the subdirectory `nls` of the folder where you installed ActivVisionTools (see section A.4 on page 56). In this file, add a line like, for example

```
AVTView.FGSelection                A File;TWAIN;MyDevice
```

Instead of `MyDevice`, insert the name of your HALCON image acquisition interface as you would specify it in a call to the HALCON operator `open_framegrabber`. If the line already exists in the file, modify it accordingly.

② **Content of** `AVTViewFG`

When an image acquisition device (more exactly: the corresponding HALCON image acquisition interface) is selected in the combo box `Name`, `AVTViewFG` adapts its layout and the content of the combo boxes automatically to the device. ActivVisionTools obtains the necessary information about the HALCON image acquisition interface by calling the HALCON operator `info_framegrabber`. Thus, to achieve that `AVTViewFG` adapts itself to your image acquisition interface, you must assure that your fimage acquisition interface returns the necessary information in `info_framegrabber`.

Table A.3 shows which query via `info_framegrabber` influences which element of `AVTViewFG`. Note that if the call to `info_framegrabber` returns an empty list or an error, the corresponding element is disabled! Thus, `info_framegrabber` should at least return a single value to enable the element. It is not necessary to return a complete list of values for combo box elements, because you can enter new values by typing them into the text field. The first list element becomes the default value.

For information about how to extend your HALCON image acquisition interface so that it returns the necessary information `info_framegrabber`, please consult the HALCON Image Acquisition Interface Programmer's Manual or contact your distributor.

③ **Additional parameters**

For a list of available parameters, please refer to the online documentation of the HALCON image acquisition interfaces, which can be found under `http://www.halcon.com/image-acquisition`. Parameters listed in the section *Parameters for open_framegrabber* are set in `AVTViewFG`, those listed in the section *Parameters for set_framegrabber_param* in `AVTViewFGParameter`. Note that ActivVisionTools and the HALCON image acquisition interfaces use a slightly different terminology.

query parameter in `info_framegrabber`	element of `AVTViewFG`	additional information
`'device'`	`Board`	
`'port'`	`Port`	
`'camera_type'`	`CamType`	If a file selection box is to appear, return `'CAMFILE:'` in the first tuple entry, a list of file extensions (without `'.'`, separated by `';'`) in the second, and a default directory in the third. Additional values for the combo box can follow.
`'color_space'` `'bits_per_channel'`	`Color`	The two queries are combined as follows: If `'color_space' = 'gray'` or `'rgb'`, the values for `'bits_per_channel'` are automatically added to form entries like `'Gray 8'` or `'Color 24'` (with NLS support). For other values of `'color_space'`, the text is directly inserted in the combo box, without NLS support; in this case, `'bits_per_channel'` is ignored; instead, when connecting to the image acquisition device, `'-1'` is passed for this parameter.
`'field'`	`Field`	
`'external_trigger'`	`Trigger`	
	`Size`	The combo box automatically contains the entries `'Full Size'`, `'Half Size'`, and `'Half Height'` for unknown image acquisition interfaces.
`'info_boards'`	`Board`, `Port`	This is a special query for dynamic information, If it returns values that are not included in the lists returned by `'device'` and `'port'`, they are inserted in the combo boxes (and used as default values).

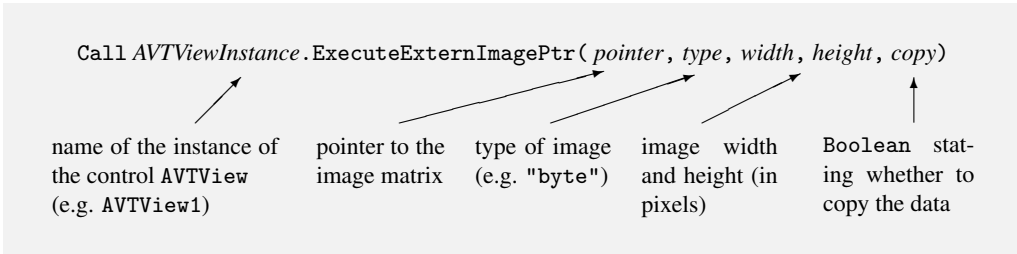
Table A.3: How queries via `info_framegrabber` influence the elements of AVTViewFG.

A.6.2 Integrating Images Directly

Via the programming interface of ActivVisionTools, you can let it process images from arbitrary sources. First, we describe the simplest way to integrate and process gray value images. After this, we briefly describe alternative methods.

Integrating Images With `ExecuteExternImagePtr`

To "feed" your own gray value images to an ActivVisionTools application, the control `AVTView` provides the method `ExecuteExternImagePtr` which is invoked (within Visual Basic) as follows:


Call AVTViewInstance.ExecuteExternImagePtr(pointer, type, width, height, copy)
name of the instance of the control AVTView (e.g. AVTView1)
pointer to the image matrix
type of image (e.g. "byte")
image width and height (in pixels)
Boolean stating whether to copy the data

Some words about the parameters: The parameter `type` specifies how a pixel is represented in memory; for a standard 8-bit gray value image specify `"byte"`, if more than 8 bits are used for each pixel specify `"uint2"`.

With the parameter `copy` you can influence whether ActivVisionTools creates a copy of the image matrix or not. Choosing `True` is the somewhat slower but definitively safer method. By choosing `False` you can speed up the application a bit, but now you must guarantee the validity of images, i.e., you must guarantee that the memory area pointed to contains the image throughout its use within the ActivVisionTools application. If you want to use external images when calibrating your vision system using ActivGeoCalib, please note that all calibration images must be available throughout the calibration process.

The ActivVisionTools distribution includes the example Visual Basic project `externimage\view_externimage.vbp` which is explained in the following. When the form is loaded, i.e. within the method `Form_Load`, three test images are created as byte arrays and "filled" with different twodimensional gray value ramps.

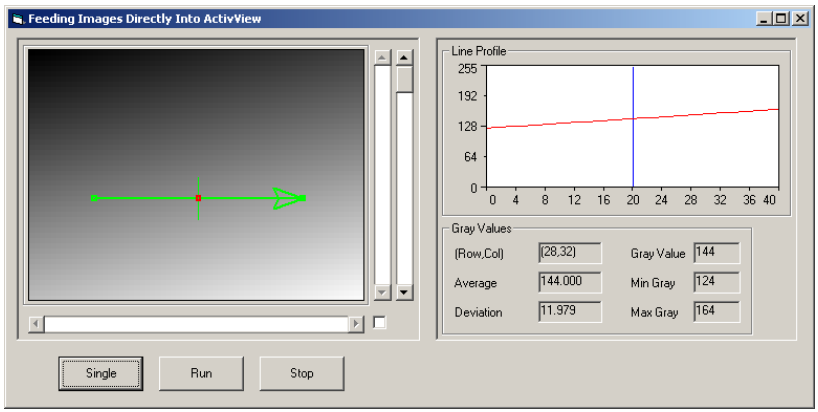


Figure A.4: Integrating external images into an ActivVisionTools application.

```
Private Image1(63, 47) As Byte, Image2(63, 47) As Byte, Image3(63, 47) As Byte

Private Sub Form_Load()

  Dim iRowCnt, iColCnt As Integer

  For iRowCnt = 0 To 47
    For iColCnt = 0 To 63
      Image1(iColCnt, iRowCnt) = CByte(iRowCnt * 4 + iColCnt)
      Image2(iColCnt, iRowCnt) = CByte(iRowCnt * 2 + iColCnt * 2)
      Image3(iColCnt, iRowCnt) = CByte(iRowCnt + iColCnt * 3)
    Next
  Next

End Sub
```

The following method returns the pointer to an image matrix, alternating between the three images.

```
Private Function NextExternImagePtr() As Long

  Static iNextImage As Integer

  Select Case iNextImage
    Case 0
      NextExternImagePtr = VarPtr(Image1(0, 0))
      iNextImage = 1
    Case 1
      NextExternImagePtr = VarPtr(Image2(0, 0))
      iNextImage = 2
    Case 2
      NextExternImagePtr = VarPtr(Image3(0, 0))
      iNextImage = 0
  End Select

End Function
```

When you now click the button `Single` (see figure A.4), the method `SingleButton_Click` is executed, which gets the pointer to an image matrix and calls `ExecuteExternImagePtr`. Besides making the image available to ActivVisionTools, this method triggers a full execution cycle. In the example project, the gray value profile along one line is calculated and displayed using ActivLineProfile (see section 3.2 on page 30).

```
Private pExternImage As Long

Private Sub SingleButton_Click()

  pExternImage = NextExternImagePtr
  Call AVTView1.ExecuteExternImagePtr(pExternImage, "byte", 64, 48, True)
  bDoRun = False
End Sub
```

To integrate images in a continuous loop, one has to take care that events are processed after each execution cycle by adding the line `DoEvents`. Otherwise, you could not even stop the loop by clicking on the button `Stop`.

```
Private bDoRun As Boolean

Private Sub RunButton_Click()

  bDoRun = True
  Do
    pExternImage = NextExternImagePtr
    Call AVTView1.ExecuteExternImagePtr(pExternImage, "byte", 64, 48, True)
    DoEvents
  Loop While bDoRun = True

End Sub
```

```
Private Sub StopButton_Click()
  bDoRun = False
End Sub
```

The example project contains additional code which in the continuous mode waits half a second after each execution cycle.

Other Methods for Integrating Images

The programming interface of ActivVisionTools provides additional methods for integrating and accessing images. They are listed and described briefly below; please refer to the **Reference Manual** for details about their signatures.

- **Integrate and process a HALCON image**

  → `ExecuteExternImage`
  This method has the same behavior as `ExecuteExternImagePtr` but expects a HALCON image (which can also be a color image!) as input.

- **Integrate an image without processing**

  → `SetHalconImagePtr`
  This method allows you to integrate gray value images from arbitrary sources by passing a pointer to it, similarly to `ExecuteExternImagePtr`. To process it, call `ExecuteAllToolsOnce`, passing `'FALSE'`for the parameter that controls whether a new image is grabbed.

  → `HalconImage`
  This method lets you integrate a HALCON image (which can also be a color image!) into your ActivVisionTools application. To process the image, call `ExecuteAllToolsOnce` as descibed above.

- **Integrate an RGB image without processing**

  → `SetHalconImageRGBPtr`
  This method works similar to `SetHalconImagePtr`, but allows to integrate RGB images from arbitrary sources by passing three pointers to the three channels. How to process the image is described above.

  → `SetHalconImageRGB`
  With this method, you can integrate an RGB image in form of three HALCON images corresponding to the three channels. How to process it is described above.

- **Access the image**

  You can access the currently used image in form of a HALCON image (→ `HalconImage`) or directly access the image pointer (→ `GetHalconImagePtr`). Note that the latter should be used for gray value images; for color images it returns the pointer to the first channel.

- **Access an image channel**

  To access a single channel of the current image, we recommend to first check the number of channel using the method `HalconImageChannelCount`. Then, you can either get the pointer to a channel (→ `GetHalconImageChannelPtr`) or access it in form of a HALCON image (→ `HalconImageChannel`).